Improvements to quantum search, with applications to cryptanalysis

Pring, Ben

Award date:
2019

Awarding institution:
University of Bath

Improvements to quantum search, with applications to cryptanalysis

submitted by

Benjamin Iain Pring

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Computer Science

January 2019

Summary

The quadratic advantage in query-complexity that Grover's algorithm offers for the unstructured search problem is well-known by the cryptographic community and the security estimates for many cryptosystems are based upon quantum resource estimates for Grover's algorithm. In essence, Grover's algorithm consists of the repeated application of a quantum oracle, unique to the instance of the search problem, and some less costly circuitry. This quantum oracle is essentially a reversible boolean circuit, implemented via quantum gates. This thesis examines the optimisation of quantum search routines in the noise-free quantum circuit model of computation, with the goal of negating a portion of the cost associated with the necessity to implement quantum oracles in Grover's algorithm. The importance of such gains is illustrated by considering the approximate circuit-size of Grover's algorithm for the single-target search problem, $\frac{\pi}{4} \cdot 2^{\frac{n}{2}} \cdot \text{poly}(n)$ where $\text{poly}(n)$ is the cost of implementing the quantum oracle. If $n = 100$ and $\text{poly}(n) = n^3$, then the approximate quantum circuit-size is 2^{70} compared to the lower bound implied by query-complexity of 2^{50} and this overhead of 2^{20} has an associated real-world cost in terms of the number of quantum gates which must be implemented. Reduction of this overhead can therefore result in gains for any potential implementation. Optimisation of the reversible circuitry which implements the quantum oracle can achieve this and this thesis demonstrates that this can also be achieved by other methods for certain problems by using modifications of Grover's algorithm. As a motivating example throughout this thesis, we apply these techniques to lower quantum resource estimates for cryptanalysis of the *Multivariate Quadratic problem* over $\mathbb{F}_2$ and later in cryptanalysis of the *Advanced Encryption Standard* (AES).

This thesis is dedicated to Janet, Iain and Matthew Pring.

It is also dedicated to walking up to lampposts and back.

"Y'all make this - - - - up, don't you?"

— Customs officer (laughing) upon reading the research topic on my visa (2019).
He wasn't wrong.

Foreword

This thesis could not have been accomplished alone and some words of thanks are in order, though words can never repay the opportunities and time afforded me by these people and organisations.

This work was funded by the EPRSC grant EP/M50645X/1 — without this source of funding, I would not have had the opportunity to pursue a PhD and I extend my thanks to the Engineering and Physical Research Science Council. Throughout my PhD, countless workshops, conferences and summer schools have provided either free or heavily reduced rates for me as a PhD student — these opportunities have given me chances to experience international research and learn about research culture around the world. In particular, I would like to thank the organisers of Real World Cryptography, the Marktoberdorf summer school, CROSSING winter school on quantum security and the PQCRYPTO summer school, all of which exposed me to a wealth of ideas and people at times when I needed it. I also thank the Statistics and Operational Research institute of the University of Lancaster for the opportunity to experience research as an undergraduate.

My family has been immensely supportive and in many ways this thesis is as much their work behind the scenes as it is my own at the lab. I cannot thank you enough. To my brother Matthew Pring, for watching films with me when I needed to relax. To my father, Iain Pring for helping with administration and to my mother Jan Pring, for simply everything (including mathematically themed cakes). My thanks to Bridget Smith, Ruth Smith and Frank Smith for their kindness and a place to stay. I would like to thank friends in Bath who have shared their company and kindness, even when deadlines and exhaustion made it difficult to reciprocate. In particular, Alice Root-Gutteridge and Dave Clipson for their excellent cooking and company, Joanna Tarko for reminding me of the pleasure that comes from reading, John Benardis for listening and laughing, Denise Lengyel and Swen Gaudl for mini-breaks and mini-biscuits and Shridhar Ravikumar, José Serra and Dmitri Kit for training me to forever associate pubs with machine learning. My sincere thanks to the Computer Science department and staff at the University of Bath and to Fran Burstall for fundamentally changing the way I viewed mathematics.

In particular, my three supervisors deserve special thanks. My thanks to John Power, who gave me my first taste of research during undergraduate years and has been kind enough to share his experiences of research culture and academia. My thanks also to Christophe Petit, who gave his time generously and fostered my ability to develop ideas. I apologise for the English humour.

Ultimately though, this PhD would neither have been started or completed without the interest, kindness and sheer force of personality of my primary supervisor, James Davenport. James' ability to be there for people, in particular his students, is unsurpassed and something I have personally benefited from time and time again. James has been patient, generous and has always had and has always made time for me, even when his own was (as always) extremely pushed. It has honestly been a pleasure being your student.

Contents

Chapter 1

Introduction

> *We may construct our cipher in such a way that breaking it is equivalent to (or requires at some point in the process) the solution of some problem known to be laborious. Thus, if we could show that solving a certain system requires at least as much work as solving a system of simultaneous equations in a large number of unknowns, of a complex type, then we would have a lower bound of sorts for the work characteristic.*
>
> - Claude Shannon (1949), *Communication theory of secrecy systems [Sha49]*

1.1 Introductory remarks

The goal of this thesis is to provide evidence that the projected resources required to solve important cryptographic problems via quantum search routines are lower than current projections may indicate, at least in the logical quantum-circuit model of computation. This stems from the fact that many *quantum resource estimates* for important problems are extrapolated by using *Grover's quantum search algorithm* (see Theorem 3.10). This is somewhat natural, as (because it assumes no structure in the problem) Grover's quantum search algorithm can be treated as a black-box algorithm whereby we can derive the cost using a standard formula and the design of a reversible boolean circuit that is particular to the search problem we are considering. Our results rely upon *amplitude amplification*, which is a generalisation of Grover's algorithm and the identification of various types of structure in the problems we consider.

Cryptography is by necessity a cross-disciplinary field and it can take time for knowledge to diffuse between different fields. The formulation of theorems in an accessible manner is an important first step in this dissemination of knowledge and in this thesis we work towards a level that is accessible by anyone with a computer science background as soon as possible. This is not a thesis about quantum computing, but a thesis on how to *use* quantum algorithms — specifically the important quantum subroutine of *amplitude amplification* (see Theorem 3.6) in order to optimise quantum search. With this in mind, we review the contents of this thesis.

It is first worth briefly sketching the problem we are examining, before formally defining the terms and algorithms. Say we are looking for a single element out of 2^n possible elements and that we only have the ability to test each element and check if it is suitable. Classically, the best we can do is test, either exhaustively or at random, each element $x \in \{0,1\}^n$ until we have found this element. Such an approach requires 2^n tests to obtain the element we are looking for in the worst-case and approximately $\frac{1}{2} \cdot 2^n$ tests on average [Ahl14].

In comparison, Grover's quantum search algorithm allows us to embed the same circuit that performs this test within a quantum circuit and after $\left\lfloor \frac{\pi}{4} \cdot 2^{n/2} \right\rfloor$ serial executions of this circuit — in addition to some other minor circuitry — allows us to obtain the element we are looking for with near certainty. This so-called quadratic speedup is clearly an advantage, but the test has an associated cost so that the full cost (which can be taken to be either the number of quantum gates or the time taken) of Grover's quantum search algorithm is in reality

$$\left\lfloor \frac{\pi}{4} \cdot 2^{n/2} \right\rfloor \cdot poly(n), \tag{1.1}$$

where $poly(n)$ is the cost of implementing the test and minor circuitry as a quantum circuit. The embedding of the classical circuit in quantum circuitry is a relatively simple procedure (though care must be taken as the circuits must be converted to *reversible boolean circuits*, which destroy no information). Quantum resource estimations for various cryptographic problems can therefore be performed simply deriving the cost of a reversible boolean circuit which performs the required test, adding the cost of the minor circuitry to derive $poly(n)$ and using it in conjunction with Equation (1.1).

Whilst there exist quantum search algorithms that exploit structure in problems to reduce the term $\left\lfloor \frac{\pi}{4} \cdot 2^{n/2} \right\rfloor$ in (1.1) [CGW00], known as the *query-complexity* term, the aim of this thesis is to examine and exploit alternative structure, so that we can reduce the $poly(n)$ overhead. That this is important, both in regards to cryptographic functions and industrial uses of quantum search is easily illustrated by the real-world cost of this overhead. If $n = 100$ and $poly(n) = n^3$, then we will have that this overhead is approximately 2^{20}. This is a substantial real-world cost, even discounting the notion that quantum computers are expected to be magnitudes more expensive than classical computers to both build and run.

Our results will stand in the *logical quantum-circuit model of computation*, which does not factor in the effect of physical noise, the cost of quantum-error correction schemes, the cost of building or running a quantum computer or the network topology of the connections between quantum bits. The important metrics will be the same as for classical circuits, in that we will wish to compute the requirements of quantum algorithms in terms of circuit-width (number of quantum bits), circuit-size (number of quantum gates) and circuit-depth (number of timesteps taken). We now review the chapters of this thesis.

In Chapter 1 we define and review various definitions of search, most importantly the *preimage search problem*, we define and provide the motivating examples we use throughout this thesis — that of the *multivariate quadratic* ($\mathcal{MQ}$) problem and the *key-search* problem.

In Chapter 2 we review the basic concepts of quantum computing required to understand the algorithms in this thesis and discuss the implementation details of *quantum phase oracles*. Our aim is not to provide an exhaustive knowledge of the intricacies of quantum computing, but to draw a straight-line towards the theory of amplitude amplification. We define quantum phase oracles (an integral component of amplitude amplification) and demonstrate several abstract methods by which they may be implemented for the preimage search problem and a constraint-based formulation of the search problem, paying attention to the explicit costs involved. We conclude by demonstrating Schwabe and Westerbaan's approach to the design of a quantum bit oracle for the $\mathcal{MQ}$ problem [SW16], the quantum search oracle for the key-search problem for AES [GLRS16] and how they fit into the abstract methods we have described.

In Chapter 3 we provide the theory and costs of quantum amplitude amplification and discuss various applications of this quantum subroutine, including Grover's quantum search algorithm. We conclude with an examination of the costs involved in using Grover's algorithm in conjunction with the quantum bit oracles for the $\mathcal{MQ}$ and AES key-search problems described in Chapter 2.

In Chapter 4 we discuss the balancing of costs in Grover's search algorithm when there exists structure that we can exploit via an *Efficient Neighbourhood Transition Strategy* — a method that can be used to evaluate a function for a reduced cost, if we have already evaluated a single point in the domain. We apply this to the $\mathcal{MQ}$ problem and derive improvements for existing quantum search methods [SW16] applied to this problem. We discuss a state-based decomposition of the quantum phase oracle that allows us to implement the ENTS strategy described in Chapter 4 in a generic manner and note that this can also be applied to the $\mathcal{MQ}$ problem.

In Chapter 5 we discuss how principles in Chapter 4 can be enhanced via preprocessing and amplitude amplification and apply this to the $\mathcal{MQ}$ problem.

In Chapter 6 examine an existing method [KYYLHH15] that exploits structure in search problems to provide lower quantum resource estimates. We examine the expected performance of this existing solution and demonstrate that it can fail in real-world scenarios as it is dependent upon a promise on the size of a subset defined by a function. We demonstrate how we can modify this procedure to restore its correctness and apply our new method to achieve new quantum resource estimates for quantum search applied to both the $\mathcal{MQ}$ and AES key-search problem which are optimised towards requiring few qubits.

In Chapter 7 we summarise and give our conclusions. In particular we discuss the impact of this work upon choosing cryptographic parameters.

1.2 Search problems

One of the simplest and most generic search problems that one can consider is that of the *unstructured search problem*. Simply put, we are searching for a length n bitstring and have only one means of identifying this element. No other information about how we could construct a more efficient algorithm is available. We will assume that we know the number of such bitstrings.

Definition 1.1 (The unstructured search problem)**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $M_\chi = |\chi^{-1}(1)|$. The *unstructured search problem* is to find an $x \in \{0,1\}^n$ such that $\chi(x) = 1$, given only the ability to evaluate χ.
The function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ will be referred to as the *boolean indicator function* for the search problem.

It is this problem that Grover's quantum search algorithm (see Theorem 3.10) solves and, given the ease by which other formulations of search problem reduce to unstructured search, gives rise to its utility in many aspects of computer science. Whilst Shor's algorithm provides a superpolynomial speedup for a narrow subset of problems with a periodic structure [Sho99] over the best known classical algorithms [LLMP93], Grover's algorithm provides only a polynomial speedup compared to a classical brute-force or exhaustive search algorithm. However, Shor's algorithm can only be applied to a narrow subset of problems whilst Grover's utility is generic and wide-ranging as search is a key-component, subroutine and bottleneck of many classical algorithms. We will primarily be interested in applying quantum search procedures to find the *preimage* of pseudorandom functions (functions that may be assumed to be uniformly sampled from the set of functions from length n bitstrings to length m bitstrings).

Definition 1.2 (The preimage search problem)**.**
Let $h : \{0,1\}^n \longrightarrow \{0,1\}^m$, $Y_h \subseteq \{0,1\}^m$ and $M_h = |h^{-1}(Y_h)|$. The *preimage search problem* is to find an $x \in \{0,1\}^n$ such that $h(x) \in Y_h$ or prove that no such element exists.
The case where $|Y_h| = 1$ will be known as the *single-target preimage search problem*, whereas the case where $|Y_h| > 1$ will be known as the *multi-target preimage search problem.*

We will predominantly concern ourselves with the single-target preimage search problem in this thesis, but the topic of the costs involved with a naive implementation of multi-target search will later come into our discussion. It is easily seen that the preimage search problem reduces to the unstructured search problem as we can create the function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ such that

$$\chi(x) \mapsto \begin{cases} 1 & \text{if } \ h(x) \in Y_h \\ 0 & \text{otherwise.} \end{cases} \tag{1.2}$$

The boolean indicator function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ for the preimage search problem is therefore simply an evaluation of h followed by a check for set membership. In the case of the single-target preimage search problem, this is a single bitstring comparison, but more efficient methods could be used if $|Y_h|$ is large. The total cost in terms of bit operations for performing a classical search procedure to solve the preimage search problem is therefore the product of the number of times χ must be evaluated, multiplied by number of bit operations that the evaluation of χ requires.

If all we have is the ability to execute the pseudorandom function $h : \{0,1\}^n \longrightarrow \{0,1\}^m$, then best we can do is to sample in some order $x_1, \ldots, x_{2^n} \in \{0,1\}^n$ and test these elements in turn by evaluating them and testing whether $h(x_i) \in Y_h$. If we consider the single-target preimage search problem then if h is pseudorandom, we have that the expected number of tests until we achieve a single success is $\frac{2^n+1}{M_h+1}$ [Ahl14]. By the above discussion, the expected number of bit operations to perform a naive *brute-force* or *exhaustive search* for the unstructured search problem is

$$\frac{2^n+1}{M_\chi+1} \cdot \Big(\text{Cost of evaluating } \chi(x)\Big) \tag{1.3}$$

which for the preimage search problem is

$$\frac{2^n+1}{M_h+1} \cdot \Big(\text{Cost of evaluating } h(x) + \text{Cost of testing if } h(x) \in Y_h\Big). \tag{1.4}$$

It is worth noting that the process of testing these evaluations is embarrassingly parallel, in that if the expected time for the above procedure to result in success is T to complete on one processor and we have P processors available, then the expected time if we use all P processors is $\frac{T}{P}$ and the number of bit operations required will remain identical.

In comparison, the cost to execute Grover's quantum search algorithm in terms of the number of quantum gates (as opposed to bit operations) and solve the unstructured search problem with probability at least $1 - \frac{M_\chi}{2^n}$ will be (see Theorem 3.10)

$$\left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{2^{n/2}}{M_\chi}} \right\rfloor \cdot \Big(\text{Cost of evaluating } \mathcal{O}_\chi^{(b)} + \text{Cost of the diffusion step on } n \text{ qubits}\Big), \tag{1.5}$$

where $\mathcal{O}_\chi^{(b)}$ is a quantum bit oracle (see Definition 2.11) which is a quantum circuit that essentially performs the quantum equivalent of evaluating χ. The cost of the diffusion step on n qubits will cost $O(n)$ quantum gates and $\text{Cost}(O_\chi^{(b)})$ will usually be $O(n^d)$ quantum gates, where $d \geq 1$.

Full details of Grover's quantum search algorithm can be found in Section 3.2.1, but for now we simply note that Grover's algorithm requires a long-running serial quantum computation and is not embarrassingly parallel. It has been proven that in relation to the unstructured search problem, the optimal parallelism strategy to follow is to essentially partition the search space into P subsets of equal size and execute Grover's algorithm upon each one. Whilst the time taken will be reduced by a factor of $\frac{1}{\sqrt{P}}$ by this strategy, the number of quantum gates required will be increased by a factor of $\sqrt{P}$ [Zal99].

Structure in either the problem or the circuit χ can help decrease the complexity of search both in terms of classical search algorithms and quantum search algorithms. For example, if we know that the solutions are not distributed uniformly throughout the domain we may choose to sample elements and evaluate them in a specific order. Alternatively, we may be able to rule out that the solution lies in large areas of the search space by performing specific tests — a process known as *pruning* the search tree.

We give an example of structure that can be exploited by both a classical search procedure and a quantum search procedure from [CGW00]. If $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ can be broken down into several tests, some of which involve a smaller set of variables so that

$$\chi(x_1 \dots x_k x_{k+1} \dots x_n) \mapsto \chi_1(x_1 \dots x_k) \wedge \chi_2(x_1 \dots x_k x_{k+1} \dots x_n) \tag{1.6}$$

then we can simply perform a search on the first k variables until we find a *partial solution* $x_1 \dots x_k \in \{0,1\}^k$ such that $\chi(x_1 \dots x_k) = 1$, then fix these variables and attempt to extend this partial solution to a full solution using a search on the remaining $n-k$ variables until we have located an assignment $x_{k+1} \dots x_n \in \{0,1\}^{n-k}$ such that $\chi_2(x_{k+1} \dots x_n) = 1$, thus providing an element $x_1 \dots x_n \in \{0,1\}^n$ such that $\chi(x_1 \dots x_n) = 1$. The expected complexity of this approach is dependent upon $M_\chi = |\chi^{-1}(1)|$ and the expected number of preimages $M_{\chi_1} = |\chi_1^{-1}(1)|$.

If we sample $x_1 \dots x_k \in \{0,1\}^k$ until we have a partial solution such that $\chi_1(x_1 \dots x_k) = 1$ and then exhaustively search the remaining $n-k$ variables in an attempt to extend this to a full solution, then this process will clearly terminate with a solution, if one exists. If we find that $\chi_1(x_1 \dots x_k) = 0$, we need not check whether $x_1 \dots x_k$ can be extended to a full solution, hence this additional structure can be used to prune the search space.

The probability of obtaining a partial solution is $\frac{M_{\chi_1}}{2^k}$, therefore we need on the order of $\frac{2^k}{M_{\chi_1}}$ trials to find a partial solution. The probability that this partial solution can be extended to a full solution is simply $\frac{M_\chi}{M_{\chi_1}}$ and, using the upper-bound 2^{n-k} for the number of evaluations we must make for χ_2, hence the number of bit operations we expect to require is

$$\frac{M_{\chi_1}}{M_\chi} \cdot \left(\frac{2^k}{M_{\chi_1}} \cdot (\text{Cost}(\chi_1)) + 2^{n-k} \cdot (\text{Cost}(\chi_2)) \right) \tag{1.7}$$

If we assume that $n > k > a > 0$, $\text{Cost}(\chi_1) = k^2$ $\text{Cost}(\chi_2) = n^2$, $M_\chi = 1$ and $M_{\chi_1} = 2^{k-a}$ then we have that a naive exhaustive search approach would require an expected

$$\frac{2^n + 1}{2} \cdot 2n^2 = (2^n + 1) \cdot n^2 \tag{1.8}$$

operations to find a solution, whilst we would expect that the nested approach to require

$$2^{k-a} \cdot \left(\frac{2^k}{2^{k-a}} k^2 + 2^{n-k} n^2 \right) = 2^k k^2 + 2^{n-a} n^2 \tag{1.9}$$

bit operations. If $k = n - a$, then the naive search requires on the order of $O(2^n \cdot n^2)$ bit operations whilst the nested search procedure requires on the order of $O(2^{n-a} \cdot n^2)$ bit operations and is efficient if $a > 2$. This can be extended to a nested approach, if the problem admits such a decomposition, and the authors of [CGW00] propose a *nested quantum search algorithm* based upon this approach which requires $O(\sqrt{2^{dn}} \cdot poly(n))$ quantum gates compared to the same classical algorithm which would require $O(2^{dn} \cdot poly(n))$ classical bit operations, where $0 < d < 1$. We discuss this quantum algorithm in Section 5.3 and will exploit a similar decomposition in Chapter 6, where the decomposition is instead of the form $\chi_1, \chi_2 : \{0,1\}^n \longrightarrow \{0,1\}$ and again $\text{Cost}(\chi_1) < \text{Cost}(\chi_2)$, but we have that both χ_1 and χ_2 involve n variables.

We will examine several types of structure within this thesis that can be used to reduce the overall cost of implementing Grover's quantum search algorithm and to reduce the overhead involved with cost of evaluation of $\mathcal{O}_\chi^{(b)}$ in Equation (1.5). We now proceed to describe the main examples of the preimage search problem that we use to illustrate the gains throughout this thesis — that of the *Multivariate Quadratic* problem over $\mathbb{F}_2$ and the *key-search problem* for symmetric-key encryption systems, specifically the Advanced Encryption Standard (AES-$\{128, 192, 256\}$).

1.3 The Multivariate Quadratic ($\mathcal{MQ}$) problem

The Multivariate Quadratic problem over the finite field $\mathbb{F}_2$ will be used to demonstrate the computational gains of all methods in this thesis. It is a problem that exhibits a large amount of structure and that maps naturally to both classical and quantum hardware, hence is a useful research tool for both investigating and benchmarking the optimisation of quantum search algorithms. This is not a thesis directly about the $\mathcal{MQ}$ problem, but in this section we survey the problem, paying special attention to the classical search method of Fast Exhaustive Search (FES) in Section 1.3.4.2, which will later play a role in Chapter 4.

1.3.1 Definitions and motivation

The *Multivariate Quadratic* ($\mathcal{MQ}$) problem is the problem of solving a system of degree-two equations over a finite field and has a role in both the construction of cryptographic schemes and in cryptanalysis. We will be concerned with cryptanalysis via quantum search in this thesis.

Definition 1.3 (The Multivariate Quadratic ($\mathcal{MQ}$ or $\mathcal{MQ}(\mathbb{F}_q, n, m)$) problem)**.**
Let $\mathbb{F}_q$ be a finite field of size q and $f^{(1)}, \ldots, f^{(m)} \in \mathbb{F}_q[x_1, \ldots, x_n]$ such that each $f^{(i)}$ is of total degree two. The *Multivariate Quadratic problem* is the problem of finding a solution vector $\bar{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ such that $f^{(1)}(x_1, \ldots, x_n) = \cdots = f^{(m)}(x_1, \ldots, x_m) = 0$.

We will refer to the general problem by the shorthand $\mathcal{MQ}$ and specific instances of the $\mathcal{MQ}$ problem via the shorthand $\mathcal{MQ}(\mathbb{F}_q, n, m)$. In this thesis will be almost entirely concerned with the case where $q = 2$ and most interesting in the case where $n = m$, which is thought to be the hardest case. We note that each $f^{(k)}(x_1, \ldots, x_n)$ which describes an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ can be represented as

$$f^{(k)}(x_1, \ldots, x_n) = \sum_{1 \leq i < j \leq n} a_{i,j}^{(k)} x_i x_j + \sum_{1 \leq i \leq n} b_i^{(k)} x_i + c^{(k)} \tag{1.10}$$

where $a_{i,j}^{(k)}, b_i^{(k)}, c^{(k)} \in \mathbb{F}_q$, owing to the fact that $x_i x_j = x_j x_i$ over any field and $x_i^2 = x_i$ over $\mathbb{F}_2$.

The field of *Multivariate Cryptography* [DGS06] is the study of public-key cryptosystems which rely upon the hardness of the $\mathcal{MQ}$ problem. These systems are often easier to solve by Gröbner bases inspired routines (see Section 1.3.5) than random $\mathcal{MQ}(\mathbb{F}_q, n, m)$ instances would be, but this structure does not help when it comes to solving them via search techniques.

We will examine one candidate multivariate quadratic public-key cryptosystem, the Gui digital signature scheme [PCY+15] in Section 1.3.7 and its proposed quantum-resistant parame-

ters [PCDY17a, PCDY17b] derived from Schwabe and Westerbaan's approach to using Grover's quantum search algorithm to solve instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ [SW16].

Performing cryptanalysis via reduction of goals to the problem of solving systems of equations over finite fields and specialised methods to solve these systems (which often possess structure and are weaker than an otherwise random system would be) has come to be known as the field of *Algebraic Cryptanalysis [Bar09a].* In the field of algebraic cryptanalysis, it has been demonstrated that it is possible to reduce the cryptanalysis of stream-ciphers, such as Trivium [DC06, TWB+14] and KeyLoq [Bar09b, CBW08] and block-ciphers (such as the Advanced Encryption Standard [Pub01, MR02]) to the problem of solving the Multivariate Quadratic problem, usually over an extension field of $\mathbb{F}_2$. The systems arising from these problems are often sparse (in that each equation involves only a small subset of the total number of variables) or contain hidden structure, making the problem of solving them very different from that of solving random $\mathcal{MQ}$ instances.

Whilst we have talked about structure arising from cryptosystems, we will concern ourselves with the hardness of solving random instances of $\mathcal{MQ}(\mathbb{F}_2, n, n)$, which is thought to be the hardest case to solve, as a case-study. We discuss this further in Section 1.3.3.

1.3.2 Why quadratic?

Before reviewing techniques to solve instances of the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ problem and the Gui digital signature scheme, whose security relies upon the hardness of solving this problem, it is worth noting the reason why the problem is stated in terms of *quadratic* equations. This stems from both cryptanalysis and the field of multivariate cryptography.

The public-key of a multivariate cryptosystem is a system of multivariate quadratic equations over a finite field and the problem of breaking such a cryptosystem is often simply equivalent to an instance of the $\mathcal{MQ}$ problem. This will need to be both stored and processed by users, hence size is an issue. As the number of unique monomials of a degree-d system (assuming $d < q$ by the identity $x^q \equiv x$ for $x \in \mathbb{F}_q$) us that the number of bits required to represent an instance of the $\mathcal{MQ}(\mathbb{F}_q, n, m)$ problem is $m \cdot \binom{n+d}{d} \cdot (\lfloor \log_2 q \rfloor + 1)$. In the case we are predominantly interested in, that of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ instances, we have that total number of bits required to represent the system will be $m \cdot \left(\binom{n+1}{2} + 1\right)$.

In terms of cryptanalysis, there exists a simple method of reducing any degree $d > 2$ system involving n equations in m variables to an instance of the $\mathcal{MQ}$ problem involving $n' > n$ variables and $m' > m$ equations. This can be done via simple substitution, as (1.11) below demonstrates.

$$vwxyz + vwx + xyz + xy + 1 = 0 \implies \begin{cases} AB + Ax + B + xy + 1 = 0 \\ A = vw \\ B = xyz \end{cases} \tag{1.11}$$

1.3.3 Basic complexity results and assumptions concerning the $\mathcal{MQ}$ problem

It is well-known that the $\mathcal{MQ}$ problem is NP-complete [GJ79] (for an explicit and well-written reduction see p.29 of [Wol05]) and in particular, random instances of $\mathcal{MQ}(\mathbb{F}_q, n, n)$ are thought not only be hard not only in the worst-case, but also believed to be impractical to solve on average [LPT+17, JV17] owing to both a lack of any algorithm to solve the general case in time less than $O(q^{dn})$ for $0 < d < 1$ and the discussion below. As no quantum algorithm to solve any NP-hard problem has been discovered, and it is thought that quantum computers cannot solve NP-complete problems in polynomial-time, this provides the motivation behind using $\mathcal{MQ}(\mathbb{F}_q, n, m)$ instances as a quantum-resistant public-key for multivariate quadratic public-key encryption systems. We give an explicit description of the public-key digital signature Gui [PCY+15] in Section 1.3.7, which details how one such public-key is constructed.

As mentioned in Section 1.3.1, there exist many different scenarios from where instances of the $\mathcal{MQ}$ problem can arise in cryptography. These lead to a variety of different definitions and scenarios. We will primarily be interested in the case that is most generic and believed to be the hardest to solve — that of *random, dense* and *determined* systems of the $\mathcal{MQ}$ problem. We will also assume that $n \leq m \leq 2n$, for reasons that will become apparent in the discussion below.

By *random*, we mean that $a_{i,j}^{(k)}, b_i^{(k)}, c^{(k)} \in \mathbb{F}_q$ in (1.10) have a $\frac{1}{q}$ chance of being any given element in $\mathbb{F}_q$. By *dense*, we mean that each equation contains a large number of monomials and that each equation almost certainly involves all variables — a random instance will therefore also be dense for large n. There are three important cases for the relationship between the number of variables n and the number of equations m in an $\mathcal{MQ}(\mathbb{F}_q, n, m)$ instance.

An $\mathcal{MQ}$ system is *determined* if $n = m$ and this case is thought to be the hardest instance to solve. We elaborate below by considering the cases where $m < n$ and $m > n$, but intuitively it can help to consider that in the case $m < n$ there are fewer equations acting as constraints, hence there are more solutions to the problem. Alternatively if $m > n$, then there is more information available (in the form of equations) that can be computationally exploited by algorithms.

In the case where $m < n$, the system is *underdetermined*. As we can expect q^{n-m} solutions to such a system [FB09], one method has simply been to choose $n-m$ variables and assign them random values. The problem is then reduced to that of solving an instance of $\mathcal{MQ}(\mathbb{F}_q, m, m)$, which is expected to possess a single solution. There exists an alternative method in the form of a reduction for underdetermined instances of the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ problem. If we have that $n = \omega m$ for some $\omega \in \mathbb{Q}_{>1}$, then Thomae and Wolf [TW12] not only give us that underdetermined systems can be solved in polynomial-time if $n > m(m+1)$ but also a classical preprocessing step to obtain an instance of $\mathcal{MQ}(\mathbb{F}_2, m - \lfloor \omega \rfloor + 1, m - \lfloor \omega \rfloor + 1)$ whose solution can be transformed into the solution of the original $\mathcal{MQ}(\mathbb{F}_2, n, m)$ instance. The complexity of this approach is $O\big(m(\lfloor \omega \rfloor m)^3\big)$ which, given that the cost of our methods will be $O(2^{n/2} \cdot poly(n))$, will not impact upon the overall complexity and is a negligible serial addition to any concrete resource estimate of the costs involved. It is therefore possible to reduce underdetermined dense systems of equations to determined systems of equations with a single solution. An instance of $\mathcal{MQ}(\mathbb{F}_q, n, m)$ is overdetermined if $m > n$. In this case Gröbner bases techniques (see Section 1.3.5) can exploit these extra equations to achieve a polynomial-time algorithm when $m > n(n+1)/2$.

1.3.4 Classical methods to solve the $\mathcal{MQ}$ problem

Classical methods to solve dense instances of the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ problem usually rely upon two ideas — search-based techniques and techniques related to the theory of Gröbner bases, with the most efficient solutions being a hybrid mixture of the two techniques. In this section we give a brief survey classical and quantum approaches to solving instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$, specifically instances of $\mathcal{MQ}(\mathbb{F}_2, n, n)$. As we are studying quantum search in this thesis, we will pay special attention to the leading method used to perform classical search — the Fast Exhaustive Search (FES) algorithm, which we later demonstrate in Chapter 4 can be used advantageously to improve the performance of Schwabe and Westerbaan's quantum search oracle for the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ problem. We will only sketch the methods employed by the other techniques.

1.3.4.1 Naive exhaustive search

For comparison, we first state the complexity of performing a naive exhaustive search procedure for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ where we treat the problem as an instance of the preimage search problem. Under our assumption that $m \geq n$ and that there exists a single solution, an $\mathcal{MQ}(\mathbb{F}_2, n, m)$ instance can easily be interpreted as the single-target preimage search problem by construction of the function $h : \{0,1\}^n \longrightarrow \{0,1\}^m$, where

$$h(x_1 \dots x_n) \mapsto f^{(1)}(x_1, \dots, x_n) \| \dots \| f^{(m)}(x_1, \dots, x_n) \tag{1.12}$$

and setting $Y_h = \{0^n\}$ and $M_h = 1$, so that $h(x)$ is simply the concatenation[1] of the result of evaluating m equations. The cost of evaluating each of the m equations then allows us to compute the total cost of a naive exhaustive single-target preimage search. Any method for evaluation of multivariate polynomials of degree two can then be employed to compute $h(x)$. There are several strategies that can now be used dependent upon the resources that are available. In the following discussion we count a single bit operation as either a multiplication or an addition.

As a basic strategy, we could simply add the constant $c^{(k)}$ to each equation register, followed by the addition of the product $x_i x_j$ if $a_{i,j}^{(k)} = 1$ and add x_i if $b_i^{(k)} = 1$. As each $f^{(k)}(x_1, \dots, x_n)$ has in the worst-case $\frac{n^2-n}{2}$ $a_{i,j}^{(k)}$ terms, n $b_i^{(k)}$ terms and one constant $c^{(k)}$, we can achieve the evaluation of each $f^{(k)}(x_1, \dots, x_n)$ by using at most $\frac{n^2-n}{2}$ multiplications and $\frac{n^2-n}{2} + n + 1$ additions. Checking if $h(x)=0^m$ requires at most m comparisons, meaning the cost of evaluating all m equations and checking them is at most $m \cdot (n^2 + 2)$ bit operations. We require at most $n+m+1$ bits of space to compute $h(x_1 \dots x_n)$ via this method, giving the single-target preimage search procedure an asymptotic complexity of $O(2^n n^2 m)$ bit operations.

If instead of computing the individual $x_i x_j$ as needed, we first precompute these values then we need only add these terms during the evaluation of each $f^{(k)}(x_1, \dots, x_n)$ via a lookup table. This means that if we possess $\frac{n^2-n}{2}$ bits of memory for storage of the $x_i x_j$ terms then we only require $\frac{n^2-n}{2} + m \cdot (\frac{n^2-n}{2} + n + 1)$ bit operations to evaluate all the equations on the same input.

[1] The notation $a \| b$ where $a = a_1 \dots a_n \in \{0,1\}^n$ and $b = b_1 \dots b_m \in \{0,1\}^m$ denotes the concatentation (or joining) of these bitstrings, so that $a \| b = a_1 \dots a_n b_1 \dots b_m \in \{0,1\}^{n+m}$.

1.3.4.2 Fast Exhaustive Search (FES)

The *Fast Exhaustive Search* (FES) algorithm [BCC+10, BCC+13] is the leading method of performing classical exhaustive search for a solution to an instance of $\mathcal{MQ}(\mathbb{F}_2, n, n)$. The FES algorithm possesses an asymptotic complexity of $O(2^{n+2} \cdot \log_2 n)$ bit operations to exhaustively enumerate all solutions to an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$, hence has the same complexity to solve the case where there exists a single satisfying solution. An open-source implementation of this algorithm is freely available at `http://www.lifl.fr/ bouillag/fes/` and the technique has been successfully applied to solve instances of the $\mathcal{MQ}$ challenge [YDH+15].

FES improves upon the naive exhaustive search approach described in Section 1.3.4.1 by improving upon two ideas. The first is relatively simple and reduces the dependence of the complexity upon m, the number of equations. If we have evaluated $k < m$ equations and found that $f^{(1)}(x_1, \ldots, x_n) = \cdots = f^{(k)}(x_1, \ldots, x_n) = 0$, but find that $f^{(k+1)}(x_1, \ldots, x_n) = 1$, then we have learnt that $h(x) \neq 0^n$ and need not continue. For any fixed $x_1 \ldots x_n \in \{0,1\}^n$, we have that the evaluation of any random $f^{(k)}(x_1, \ldots, x_n)$ modelled as a pseudorandom function has an equal probability of probability evaluating to either 0 or 1. For any candidate $x_1 \ldots x_n \in \{0,1\}^n$, the average number of evaluations we need before we learn that $h(x) \neq 0^n$ is then only 2 by the expected value of the geometric distribution. This can also be applied to the naive exhaustive search strategy discussed in Section 1.3.4.1, implying that we can reduce the cost to an expected $O(2^n n^2)$ bit operations for an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ with a single solution.

The second method by which FES improves upon naive exhaustive search is by adapting the so-called *folklore differential technique* (in that the technique is already known by the community, but without an obvious original reference), which notes that once we have evaluated a single equation upon $x_1 \ldots x_n \in \{0,1\}^n$ and obtained $f^{(k)}(x_1, \ldots, x_n)$ at a cost of $O(n^2)$ operations, we can evaluate $f^{(k)}(x_1, \ldots, x_v \oplus 1, \ldots, x_n)$ for a cost of $O(n)$ by exploiting the $\mathbb{F}_2$-*derivative*.

Definition 4.4 (The $\mathbb{F}_2$-derivative [BCC+10])
Let $f \in \mathbb{F}_2[x_1, \ldots, x_n]$. The $\mathbb{F}_2-$derivative of f with respect to x_v for $1 \leq v \leq n$ is defined as

$$\frac{\mathrm{d}f}{\mathrm{d}x_v}(x_1, \ldots, x_n) = f(x_1, \ldots, x_v \oplus 1, \ldots, x_n) \oplus f(x_1, \ldots, x_v, \ldots, x_n). \tag{1.13}$$

The Fast Exhaustive Search algorithm then exploits the fact that

$$f^{(k)}(x_1, \ldots, x_v \oplus 1, x_n) = f^{(k)}(x_1, \ldots, x_n) \oplus \frac{\mathrm{d}f^{(k)}}{\mathrm{d}x_v}(x_1, \ldots, x_n) \tag{1.14}$$

which is computationally advantageous, as if we have already evaluated and stored $f(x_1, \ldots, x_n)$ then we can evaluate a point in the domain which differs by only one bit for the cost of computing and adding $\frac{\mathrm{d}f}{\mathrm{d}x_v}(x_1, \ldots, x_n)$. The $\mathbb{F}_2-$derivative of a degree-two equation $f^{(k)}(x_1, \ldots, x_n)$ is

$$\frac{\mathrm{d}f^{(k)}}{\mathrm{d}x_v}(x_1, \ldots, x_n) = f^{(k)}(x_1, \ldots, x_n) \oplus b_v^{(k)} \oplus \sum_{\substack{i=1 \\ i \neq v}}^{n} a_{i,v}^{(k)} x_i \oplus f^{(k)}(x_1, \ldots, x_n) \tag{1.15}$$

which is an affine equation of degree one.

The $\mathbb{F}_2$-derivative $\frac{\mathrm{d}f^{(k)}}{\mathrm{d}x_v}(x_1, \ldots, x_n)$ can therefore be easily precomputed by extracting these coefficients and evaluated by using at most $n-1$ additions of x_i to add corresponding $a_{i,v}^{(k)}$ terms and a single bitflip for the $b_v^{(k)}$ term. If we consider the cost of evaluating a single equation $f^{(k)}(x_1, \ldots, x_n)$ over all of the 2^n possible inputs, then we can simply start with $f^{(k)}(0, \ldots, 0) = c^{(k)}$ and use a *Gray code* (see Section 4.3.2) to enumerate and test all 2^n elements of the domain $\{0,1\}^n$ by flipping one bit at a time (so that we update the state) and adding the $\mathbb{F}_2-$derivative $\frac{\mathrm{d}f^{(k)}}{\mathrm{d}x_v}$ on each bitflip to each equation. Used naively (and without taking account the reduction of the dependency upon m discussed earlier), this method means that we only requires $O(mn^2 + 2^n mn)$ bit operations to enumerate all solutions to an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$. We will later exploit the folklore differential technique in order to reduce the cost of quantum search applied to instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ in Chapter 4.

The Fast Exhaustive Search algorithm uses these techniques in combination with a recursive strategy for the evaluation of $\frac{\mathrm{d}f}{\mathrm{d}x_v}$ for the general case of degree-d polynomials over $\mathbb{F}_2$ to achieve a complexity of $O(d \cdot 2^n + n^{2d})$ bit operations to find the satisfying solutions for a single degree-d equation over $\mathbb{F}_2$. However, this is only for the case of one equation and unfortunately the filtering process which reduces the dependence upon m described earlier cannot be used with their technique of enumeration via Gray codes. Their solution is to determine a subset of $(\mathbb{F}_2)^n$ that satisfies a $m' \leq m$ of the polynomials and then simply test these elements on the remaining $m - m'$ polynomials to check whether these elements are satisfied, aborting the computation early once the assignment leads to an unsatisfied polynomial. The optimal value of m' to balance these costs is therefore calculated to be (after optimisation) $m' = 1 + \log_2 n$. This gives us the stated complexity of $O(2^{n+2} \log_2 n)$.

We will later exploit a quantum analogue of the strategy to reduce dependency upon m in Chapter 6 and the folklore differential technique in Chapter 4.

1.3.5 Algebraic techniques

The majority of other methods to solve instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ can be traced back to variants of algorithms which either explicitly use, or are related to methods involving *Macaulay matrices*. Essentially, given an generic instance of $\mathcal{MQ}(\mathbb{F}_q), n, m)$ one can build a matrix M_D where each column represents a monomial of degree $\leq D$ involving representing the elements of the set

$$M_D = \{\mathbf{x} \cdot f^{(k)} : k = 1, \ldots, m \text{ and } \mathbf{x} \text{ is a monomial of degree} \leq D - 2\}. \tag{1.16}$$

for a given term ordering $\prec$, where each row represents an element of the set M_D. In this way, linear algebra techniques can be used to either locate elements which directly give, or can be used to narrow the search space of solutions to the instance of $\mathcal{MQ}(\mathbb{F}_q, n, m)$. As these techniques involve manipulation of large matrices, they can be computationally intensive, even though efficient linear algebra techniques are being exploited. The Gröbner bases techniques of $F4/F5$ [Fau99, Fau02] and the XL (eXtended Linearisation) family of algorithms rely upon these techniques [AFI+04], with the XL algorithm [CKPS00, CP03] often being used in conjunction with sparse linear algebra techniques to manage the complexity of large matrix operations.

These techniques will not be used in this thesis, but it is worth noting the complexity of the leading method of solving instances $\mathcal{MQ}(\mathbb{F}_2, n, n)$ classically is the BooleanSolve algorithm [BFSS13], which uses a strategy based upon fixing $k < n$ of the variables and performing an intensive computation to check if the resulting Macaulay matrix of the $\mathcal{MQ}(\mathbb{F}_2, n-k, n)$ system has a solution — as it turns out that finding that a system does not admit a solution is faster than solving it, this proves to be efficient, though the Fast Exhaustive Search algorithm is expected to be practical for concrete parameters up until $n = 200$ [BCC+10]. The complexity of BooleanSolve is $O(2^{0.892n})$ bit operations to solve an instance of $\mathcal{MQ}(\mathbb{F}_2, n, n)$ and this technique has later been developed into the Crosbred algorithm [JV17] — the complexity of this algorithm has not yet been fully analysed at the time of writing, though practical experiments have demonstrated that it overtakes the Fast Exhaustive Search algorithm when $n = 37$.

1.3.6 Quantum methods to solve the $\mathcal{MQ}$ problem

This thesis is about the optimisation of quantum search applied to the preimage search problem and we use the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ problem to illustrate the computational gains we can make using our techniques. We briefly highlight the known quantum algorithms to solve the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ problem here in this section. The methods in this thesis are derived from studying the initial quantum resource estimation performed by Schwabe and Westerbaan [SW16] for the solution of the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ problem, which treats the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ instance as a preimage search problem whereby the classical circuit embedded into quantum circuitry and exploited by Grover's algorithm simply evaluates an element of the domain and checks if this evaluation is equal to the target image. This approach requires a quantum circuit-size of $O(2^{n/2} \cdot n^2 m)$ gates. We discuss their construction in Section 2.5.1 and provide a slightly optimised version in Section 2.5.2.

There exist alternative methods to solving an instance of the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ problem via exploiting Grover's algorithm in a manner that differs from treating the problem as a preimage search problem solved by evaluation and testing. We note that these methods were published after the author of this thesis had devised several of the techniques in this thesis. These include QuantumBooleanSolve [FHK+17] and QXL [BY18], which essentially embed the circuits used for BooleanSolve and XL discussed in Section 1.3.4 within quantum circuitry and exploit Grover's algorithm to increase the efficiency of these algorithms. A comparison of the parameters for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ where one algorithm gains a concrete advantage in circuit-complexity over another is left for future work, but for small parameters the techniques in this thesis may outperform these superior methods as their asymptotic advantage hides constant terms.

We recall that BooleanSolve performs an exhaustive search of the first of $k < n$ variables, fixing these and testing whether the resulting system has a solution. QuantumBooleanSolve replaces this exhaustive search procedure, so that the embedded classical circuit tests whether the resulting system possesses a solution or not. With this strategy, QuantumBooleanSolve requires $O(2^{0.462n})$ quantum gates to solve an instance of $\mathcal{MQ}(\mathbb{F}_2, n, n)$. GroverXL also requires $O(2^{0.462n})$ quantum gates to solve an instance of the $\mathcal{MQ}(\mathbb{F}_2, n, n)$ problem. As GroverXL essentially uses the same strategy as QuantumBooleanSolve of fixing variables and testing whether the resulting system admits a solution, the identical asymptotic complexity is not suprising.

There exists another recent algorithm in literature [CG17], which which involves the use of the HHL [HHL09] quantum algorithm for solving linear systems but whose efficiency depends upon the condition number of the corresponding boolean Macaulay matrix for the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ instance at hand. To the author's knowledge, the study of the condition number of the corresponding Macaulay matrices derived from $\mathcal{MQ}(\mathbb{F}_2, n, m)$ instances is an open problem and the results simply imply that $\mathcal{MQ}(\mathbb{F}_2, n, m)$ instances with a small condition number may be easier to solve. As far as the author is aware, this difficulty in determining the condition number of matrices means that the impact of this algorithm is still unknown.

We do not go into details in this thesis concerning these algorithms apart from Schwabe and Westerbaan's approach in Section 2.5.1, simply noting that both the quantum versions of BooleanSolve and the XL algorithm are asymptotically superior to the quantum search methods employed in this thesis, but may require a larger number of qubits for smaller instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ and that all methods are based upon applying known quantum algorithms to classical approaches to solve instances of the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ problem. Whether our methods can be used to improve the performance of QuantumBooleanSolve or QuantumXL, which simply exploit Grover's algorithm, is an open problem.

1.3.7 A motivating example: the Gui digital signature scheme

There are many varieties of Multivariate Quadratic ($\mathcal{MQ}$) cryptosystems — we highlight the construction of the Gui $\mathcal{MQ}$ digital signature [PCY+15] scheme as a motivating example and target for cryptanalysis, as the public-key of a user for the Gui signature scheme is an underdetermined system of $\mathcal{MQ}$ equations over $\mathbb{F}_2$ and forging the signature of a user corresponds to solving several instances of $\mathcal{MQ}(\mathbb{F}_2, n, n)$.

1.3.7.1 Generic Multivariate Quadratic cryptosystems

The public key of a generic $\mathcal{MQ}$ cryptosystem consists of the map $\mathcal{P} : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m$ described by an ordered $\mathcal{MQ}$ system over $\mathbb{F}_q$ and the private key consists of a method to efficiently find a preimage of $\mathcal{P}$. The generic method to construct such a system consists of three components which define the private key

- The affine (or linear) uniformly randomly chosen map of maximal rank $S : \mathbb{F}_q^m \longrightarrow \mathbb{F}_q^m$.
- The affine (or linear) uniformly randomly chosen map of maximal rank $T : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^n$.
- An $\mathcal{MQ}$ system $\mathcal{F} : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m$ whose structure allows us to compute $\mathcal{F}^{-1}$ efficiently.

The public key is then the composition of these maps

$$\mathcal{P} = S \circ \mathcal{F} \circ T. \tag{1.17}$$

As S, T and $\mathcal{F}$ are all efficiently invertible, to find a preimage of $\mathbf{h} \in (\mathbb{F}_q)^m$ the holder of the private key simply computes

- $\mathbf{x} = S^{-1}(\mathbf{h})$
- $\mathbf{y} = \mathcal{F}^{-1}(\mathbf{x})$
- $\mathbf{z} = T^{-1}(\mathbf{y})$

By composition, the holder the private-key therefore has an efficient method of computing (if one exists) the preimage $\mathbf{z} \in \mathbb{F}_q^n$ such that $\mathcal{P}(\mathbf{z}) = \mathbf{h}$.

The random affine (or linear) maps S and T hide the special structure of the map $\mathcal{F}$, which we know how to invert. Ideally, the resulting public key should both appear and act as a random instance of the $\mathcal{MQ}$ problem and contain no special structure. Unfortunately, many of the methods used to create the central map $\mathcal{F}$ lead to special structure appearing $\mathcal{P}$, which has been exploited by cryptanalysts to break the security of the system. Ultimately the security of $\mathcal{MQ}$ cryptosystems relies on two problems — *direct attacks* and *structural attacks*. Direct attacks consist of treating the public key $\mathcal{P}$ as a purely algebraic problem and attempting to solve it. Direct attacks can include those described in Section 1.3.4 and 1.3.6 for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ and in the general case for instances of $(\mathbb{F}_q, n, m)$ are performed using methods based upon Gröbner basis algorithms, exhaustive search methods or a hybrid mix of the two.

Structural attacks take into account how the central map was created, exploiting this knowledge in order to break the cryptosystem by gaining knowledge of S, T and $\mathcal{F}$ and are beyond the scope of this thesis. The focus of this thesis is on the optimisation of quantum search routines and we treat the problem as purely a preimage search problem. We do not take into account these algebraic or structural weaknesses in the public-key $\mathcal{P}$. For our purposes, $\mathcal{P}$ may as well be a random instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$, but it is worth noting that the public-key of an $\mathcal{MQ}$ cryptosystem is usually easier to solve with algebraic methods such as Gröbner bases routines than a truly random instance of $\mathcal{MQ}(\mathbb{F}_q, n, m)$ [DK12, DY13, DG10].

1.3.8 Hidden Field Equation (HFE) cryptosystems

The Hidden Field Equations (HFE) construction is one potential method of realising the invertible central map we described in construction of a generic $\mathcal{MQ}$ cryptosystem in Section 1.3.7.1. The basic construction is defined by the affine maps of maximal rank $T, S : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^n$ and the $\mathcal{MQ}$ central map $\mathcal{F} : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^n$. The efficient inversion of $\mathcal{F}$ for HFE cryptosystems is reliant upon the (so-called) canonical isomorphism $\Phi : \mathbb{F}_q^n \longrightarrow \mathbb{F}_{q^n}$ given by

$$\Phi(x_1, \ldots, x_n) \mapsto \sum_{i=1}^{n} x_i Z^{i-1} \tag{1.18}$$

where $\mathbb{F}_{q^n} = \mathbb{F}_q[Z]/(p(Z))$ and p is an irreducible polynomial of degree n. The isomorphism Φ allows the construction of an instance of $\mathcal{MQ}(\mathbb{F}_q, n, n)$ which can be efficiently inverted via first choosing a polynomial $\bar{F} \in \mathbb{F}_{q^n}[X]$ of degree D, where $X = \sum_{i=1}^{n} x_i Z^{i-1}$ of the special form

$$\bar{\mathcal{F}} = \sum_{\substack{i,j \in \mathbb{N}_0 \\ 0 \leq q^i + q^j \leq D}} A_{i,j} X^{q^i + q^j} + \sum_{\substack{i \in \mathbb{N}_0 \\ 0 \leq q^i \leq D}} B_i X^{q^i} + C. \tag{1.19}$$

As the coefficients are non-zero for only powers of $X^{q^i+q^j}$, X^{q^i} and X^0, this leads to an instance of $\mathcal{MQ}(\mathbb{F}_q, n, n)$ as if we consider the isomorphism $\Phi^{-1} : \mathbb{F}_{q^n} \longrightarrow \mathbb{F}_q^n$ applied to the term $A_{i,j} X^{q^i+q^j}$ for arbitrary $i, j \in \mathbb{N}$ then we have that

$$\Phi^{-1}(A_{i,j} X^{q^i+q^j}) = \Phi^{-1}(A_{i,j} X^{q^i} \cdot X^{q^j}) \tag{1.20}$$

$$= \Phi^{-1}\left(\left(\sum_{k=1}^{n} a_{i,j,k} Z^{k-1}\right) \cdot \left(\sum_{k=1}^{n} x_k Z^{k-1}\right)^{q^i} \cdot \left(\sum_{k=1}^{n} x_k Z^{k-1}\right)^{q^j}\right) \tag{1.21}$$

and as for $a, b \in \mathbb{F}_{q^n}$ we have that $(a+b)^{q^i} = a^{q^i} + b^{q^i}$ and for $x_i \in \mathbb{F}_q$ we have that $x_i^q = x_i$ gives

$$= \Phi^{-1}\left(\left(\sum_{k=1}^{n} a_{i,j,k} Z^{k-1}\right) \cdot \left(\sum_{k=1}^{n} x_k Z^{(k-1)\cdot q^i}\right) \cdot \left(\sum_{k=1}^{n} x_k Z^{(k-1)\cdot q^j}\right)\right) \tag{1.22}$$

If we then multiply out the argument and compute the argument modulo $P(Z)$, then we obtain

$$= \Phi^{-1}\left(\sum_{k=1}^{n}\left(\sum_{1\leq r,s\leq n} a_{r,s}^{(k)} x_r x_s\right) Z^{k-1}\right)., \tag{1.23}$$

where $a_{r,s}^{(k)}$ is simply the collected terms after reduction by $P(Z)$. Finally, we can apply Φ^{-1} directly to obtain a vector whose components are sums of quadratic terms

$$= \left(\Big(\sum_{1\leq r,s\leq n} a_{r,s}^{(1)} x_r x_s\Big), \ldots, \Big(\sum_{1\leq r,s\leq n} a_{r,s}^{(n)} x_r x_s\Big)\right). \tag{1.24}$$

In the same way we have that the isomorphism applied to each term X^{q^i} results in a vector whose components are simply linear equations in $x_1, \ldots, x_n$ and the isomorphism Φ^{-1} applied to C results in a vector of constant terms. Using the additive property of the isomorphism, we can compute the map Φ^{-1} applied to each term of the polynomial $\bar{\mathcal{F}}(X)$ and add these vectors component-wise to obtain a system of n multivariate quadratic equations in n variables over $\mathbb{F}_q$.

When it comes to inverting this system of equations, we therefore simply take the element $\mathbf{y} \in \mathbb{F}_q^n$ and apply Φ to lift it to $\mathbf{Y} \in \mathbb{F}_{q^n}[Z]/P(Z)$. Knowledge of the univariate polynomial $\bar{\mathcal{F}}(X)$, then allows $\bar{\mathcal{F}}(X)$ then allows us to compute the root of $\bar{\mathcal{F}}(X) - \mathbf{Y} = 0$ via any root-finding algorithm for finite fields and the solution $\mathbf{X} \in \mathbb{F}_{q^n}$ can then be converted to an element $\mathbf{x} \in \mathbb{F}_q^n$ using the isomorphism Φ^{-1}. After optimisations, the univariate root-finding procedure will usually be the bottleneck in this procedure and its complexity is dependent upon the degree D of the polynomial $\bar{\mathcal{F}} \in \mathbb{F}_{q^n}[X]$ — using Berlekamp's algorithm gives a complexity of $O(D^3)$ operations over $\mathbb{F}_{q^n}$ and hence choosing D to be small allows this process to be efficient.

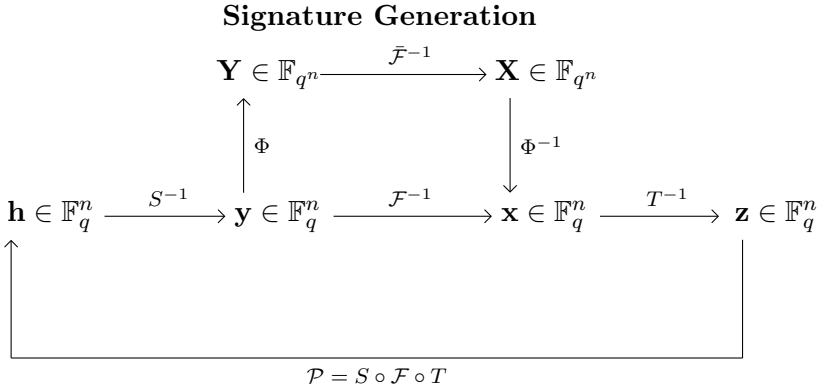


Figure 1-1: The workflow of a generic HFE cryptosystem

The hardness of solving $\mathcal{F}$ via algebraic attacks is dependent upon a parameter known as the *degree of regularity* of an $\mathcal{MQ}$ system. An upper bound for the degree of regularity of HFE systems scales with $\log_q D$, hence the degree D of $\bar{\mathcal{F}}$ must be chosen with respect to both the efficiency for the root-finding step and the security of the system against algebraic attacks.

1.3.9 Gui: An example of HFEv-

GUI [PCY+15] was introduced by Petzoldt et. al in 2015 and will be our leading example of an HFE, in fact HFEv-, multivariate signature scheme. We will examine how our methods apply to finding a preimage for the $\mathcal{MQ}$ system for the public key of this cryptosystem. To our knowledge, it remains unbroken classically, but the chosen parameters for quantum security are susceptible to our improvements to quantum search techniques for the $\mathcal{MQ}$ problem over $\mathbb{F}_2$, in Chapters 4, 5 and 6 in this thesis.

GUI [PCY+15] is an updated form of the QUARTZ HFE cryptosystem [PCG01], which takes into account theoretical advances concerning the aforementioned *degree of regularity* for HFE cryptosystems [DK12, DY13, DG10] that were not available during the design phase of QUARTZ. It uses both vinegar variables (the substitution of random field elements) and the minus technique (the removal of equations) in the central map and the base field of the resulting $\mathcal{MQ}$ system is $\mathbb{F}_2$, leading to the authors naming it after a design of Chinese pottery named Gui with three legs, representing the three design principles of HFE it implements. We will first describe the central map, which is a special case of the QUARTZ central map, where the only difference is that QUARTZ allows a finite field of arbitrary size to be used, instead of Gui's fixed choice of $\mathbb{F}_2$. Gui is parameterised by the tuple (n, D, a, v, k), where n is the number of variables, D is the degree of the central HFE map, a is the number of equations we remove, v is the number of vinegar variables in the central map and k is the number of times a preimage of the public-key is computed.

1.3.10 The HFEV- core map

The HFEv- core map is parameterised by the tuple $(\mathbb{F}_q, n, D, a, v)$ and the central map of Gui is simply an HFEv- core map with $\mathbb{F}_q$ fixed to be $\mathbb{F}_2$. We therefore describe the generic HFEv- core map and inversion procedure.

In the context of the generic HFE construction as described in Section 1.3.7.1, the affine components are the maps, $S : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^{n-a}$ and $T : \mathbb{F}_q^{n+v} \longrightarrow \mathbb{F}_q^{n+v}$ of maximal rank and the central map $\mathcal{F}$ is constructed from the canonical isomorphism $\Phi : \mathbb{F}_q^n \longrightarrow \mathbb{F}_{q^n}$ applied to the map $\bar{\mathcal{F}} : \mathbb{F}_{q^n} \times \mathbb{F}_q^v \longrightarrow \mathbb{F}_{q^n}$, defined by

$$\bar{\mathcal{F}}(X) = \sum_{0 \leq i \leq j}^{q^i+q^j \leq D} \alpha_{i,j} \cdot X^{q^i+q^j} + \sum_{i=0}^{q^i \leq D} \beta_i(v_1, \ldots, v_v) \cdot X^{q^i} + \gamma(v_1, \ldots, v_v), \tag{1.25}$$

where $\beta_i : \mathbb{F}_q^v \longrightarrow \mathbb{F}_{q^n}$ and $\gamma : \mathbb{F}_q^v \longrightarrow \mathbb{F}_{q^n}$ are respectively degree one and degree two affine maps in the vinegar variables. The central map is then defined by $\mathcal{F} : \mathbb{F}_q^{n+v} \longrightarrow \mathbb{F}_q^n$,

$$\mathcal{F} = \Phi^{-1} \circ \bar{\mathcal{F}} \circ \Phi \tag{1.26}$$

and the public-key is the composition $\mathcal{P} : \mathbb{F}_q^{n+v} \longrightarrow \mathbb{F}^{n-a}$ given by

$$\mathcal{P} = S \circ \Phi^{-1} \circ \bar{\mathcal{F}} \circ \Phi \circ T. \tag{1.27}$$

In order to avoid birthday attacks due to the chosen sizes of input, the authors of Gui use $k > 1$ applications of the HFEv- core map to sign and verify messages. In order to sign a document d, an instance of Gui(n, D, a, v, k) consists of the following process:

1. A hash of the message $\mathbf{h} \leftarrow \mathcal{H}(d)$ we wish to sign is first computed.
2. The variable $S_0 := 0^{n-a} \in \mathbb{F}_2^{n-a}$ is initialised.
3. For $i = 1, \ldots, k$:
    a. The variable D_i is set to be the first $n - a$ bits of $\mathbf{h}$.
    b. The preimage of $\mathcal{P}^{-1}(D_i \oplus S_i)$ is then computed and the result interpreted as an element $S_i \in \mathbb{F}_2^{n-a}$ and an element $X_i \in \mathbb{F}_2^{a+v}$.
    c. A hash of the original hash is then computed so that $\mathbf{h} \leftarrow \mathcal{H}(\mathbf{h})$.
4. The signature $\sigma = (S_k \| X_k \| \ldots \| X_k)$ is output.

Verification, which is unimportant for our purposes, is then handled by performing this operation in reverse, with the signature being verified if and only if the final value of $S_0 \stackrel{?}{=} 0^n \in \mathbb{F}_2^{n-a}$.

1.3.11 Gui as a target for cryptanalysis

As a target for cryptanalysis via search based methods, we have for an instance of Gui(n, D, a, v, k) that the public-key is an instance of $\mathcal{MQ}(\mathbb{F}_2, n + v, n - a)$ and that the most costly part of the process of forging a signature will be the inversion of the public-key $\mathcal{P} : \mathbb{F}_2^{n+v} \longrightarrow \mathbb{F}_2^{n-a}$ a total of k times, which must be performed in serial. Either fixing variables or using methods to solve underdetermined systems of equations [TW12] implies that the total effort involved will be that required to solve k serial instances of $\mathcal{MQ}(\mathbb{F}_2, n - a, n - a)$. We make the assumption that there exists a single solution, as do the authors of the Gui cryptosystem when choosing their quantum-resistant parameters [PCDY17a, PCDY17b].

With regards to choosing parameters for a quantum-resistant for Gui, the authors in the original design document for Gui [PCY+15] state that choosing n to be twice as large as their parameters for resistance to classical cryptanalysis will provide protection from attacks by quantum computers. This doubling of n takes into account the square-root advantage that Grover provides over classical search ($2^{n/2}$ quantum queries compared to 2^n classical queries) but does not take into account the overhead involved with implementing the quantum oracle.

After the publication of Schwabe and Westerbaan's quantum oracle (see Definition 2.11) for use with Grover's algorithm to solve instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$, the authors of Gui suggest new parameters based upon this attack being optimal — it is these targets we attack in this thesis.

Gui(n, D, a, v, k)	Security level	Cryptanalysis target	Source
Gui$(94, 17, 4, 4, 4)$	$\lambda = 80$ (classical)	$4\times\mathcal{MQ}(\mathbb{F}_2, 90, 90)$	[PCY$^+$15]
Gui$(95, 9, 5, 5, 3)$	$\lambda = 80$ (classical)	$3\times\mathcal{MQ}(\mathbb{F}_2, 90, 90)$	[PCY$^+$15]
Gui$(96, 5, 6, 6, 3)$	$\lambda = 80$ (classical)	$3\times\mathcal{MQ}(\mathbb{F}_2, 90, 90)$	[PCY$^+$15]
Gui$(127, 9, 3, 4, 4)$	$\lambda = 120$ (classical)	$4\times\mathcal{MQ}(\mathbb{F}_2, 124, 124)$	[PCY$^+$15]
Gui$(188, 17, 4, 4, 4)$	$\lambda = 80$ (quantum)	$4\times\mathcal{MQ}(\mathbb{F}_2, 184, 184)$	[PCY$^+$15]
Gui$(190, 9, 5, 5, 3)$	$\lambda = 80$ (quantum)	$3\times\mathcal{MQ}(\mathbb{F}_2, 185, 185)$	[PCY$^+$15]
Gui$(192, 5, 6, 6, 3)$	$\lambda = 80$ (quantum)	$3\times\mathcal{MQ}(\mathbb{F}_2, 186, 186)$	[PCY$^+$15]
Gui$(254, 9, 3, 4, 4)$	$\lambda = 120$ (quantum)	$4\times\mathcal{MQ}(\mathbb{F}_2, 251, 251)$	[PCY$^+$15]
Gui$(120, 9, 3, 3, 2)$	$\lambda = 80$ (quantum)	$2\times\mathcal{MQ}(\mathbb{F}_2, 117, 117)$	[PCDY17b]
Gui$(212, 9, 3, 4, 2)$	$\lambda = 128$ (quantum)	$2\times\mathcal{MQ}(\mathbb{F}_2, 209, 209)$	[PCDY17b]
Gui$(464, 9, 7, 8, 2)$	$\lambda = 256$ (quantum)	$2\times\mathcal{MQ}(\mathbb{F}_2, 457, 457)$	[PCDY17b]

Table 1.1: Suggested parameters for the Gui cryptosystem [PCY$^+$15, PCDY17a, PCDY17b].

In particular we note the case of Gui$(188, 17, 4, 4, 4)$ and Gui$(120, 9, 3, 3, 2)$ which are both proposed to possess the property that forging a signature requires at least 2^{80} quantum gates. Both parameter sets for Gui were proposed corresponding to an attack by Grover's algorithm (see Theorem 3.10) being used to search for a preimage, — but Gui$(188, 17, 4, 4, 4)$ takes into account only the number of calls to the quantum oracle that are required, whilst Gui$(120, 9, 3, 3, 2)$ takes into account both the number of calls and the cost of calling the quantum oracle. It is clear that these parameters and targets for cryptanalysis are very different and that solving Gui$(120, 9, 3, 3, 2)$ will be far easier. This provides a motivating case-study for our optimisations and leads to our conclusions in Chapter 7, that it is fundamentally dangerous to extrapolate cryptographic parameters from quantum resource estimates for quantum search algorithms — the true costs of these algorithms, even in the logical quantum circuit-model, is not yet fully understood.

1.4 The key-search problem

In this section we describe the computational resources required to attack the key-search problem for block-ciphers via search techniques, which we will examine again in Chapter 6.

Definition 1.4 (Block cipher [MKVOV96])**.**
A block-cipher with a message-block length of n and a key-length of k is defined by the *message-space* $\{0,1\}^n$, the *keyspace* $\mathcal{K} \subseteq \{0,1\}^k$, a function Gen which randomly samples a key $k \in \mathcal{K}$ and the two functions

$$\text{Enc} : \{0,1\}^n \times \{0,1\}^k \longrightarrow \{0,1\}^n \tag{1.28}$$

and

$$\text{Dec} : \{0,1\}^n \times \{0,1\}^k \longrightarrow \{0,1\}^n, \tag{1.29}$$

which must have the property that for $(P,K) \in \{0,1\}^n \times \mathcal{K}$ it holds that $\text{Dec}(\text{Enc}(P,K),K) = P$.

The usual definition of breaking a block-cipher is either the recovery of a plaintext $P \in \{0,1\}^n$, given a ciphertext $C \in \{0,1\}^n$ such that $C = \text{Enc}(P,K)$ for a given $K \in \mathcal{K}$ or the recovery of the key K used to encrypt P, which naturally implies obtaining P and the decryption of all other $P' \in \{0,1\}^n$ encrypted under this choice of $K \in \mathcal{K}$. We will be interested in the latter scenario, known as the *key-search problem for block ciphers* (see Definition 1.5).

The example to which we will immediately specialise to is that of the Advanced Encryption Standard [Pub01]. The Advanced Encryption Standard (AES) is defined with a fixed block-length of $n = 128$ and $\mathcal{K} = \{0,1\}^k$ for $k = 128, 192, 256$ and these cases will be respectively referred to as AES-128, AES-192 and AES-256. The Advanced Encryption Standard was the winning entry of the five year NIST[2] standardisation process (or competition) to choose and establish a standardised, publicly available and community-supported block-cipher from entries from the cryptographic community. The Rijndael block-cipher was eventually chosen to be the Advanced Encryption Standard and remains the standard choice of symmetric-key encryption for the majority of web traffic today.

For our purposes the implementation details of AES will not matter, only that we will model AES as a pseudorandom function in the *key-search* scenario for cryptanalysis of this primitive. The costs required to solve the key-search problem via Grover's algorithm exist in literature [GLRS16] and will be recalled explicitly in Section 2.6 to be used in Chapter 6.

Definition 1.5 (The key-search problem for block-ciphers)**.**
Let $(\text{Gen}, \text{Enc}, \text{Dec})$ define a fixed block-cipher with a message block size n and a key size k. Let $K \leftarrow \text{Gen}(1^k)$ be fixed. Let $r \in \mathbb{N}$ and $P_1, \ldots, P_r \in \{0,1\}^n$. Suppose we can obtain

$$\vec{P_r} = ((P_1, C_1), \ldots, (P_r, C_r)) \qquad \text{where} \qquad C_i = \text{Enc}(P_i, K). \tag{1.30}$$

for any choice of r and $P_1, \ldots, P_r$. The *key-search* problem is to find the fixed key $K \in \{0,1\}^k$.

[2]National Institute of Standards and Technology (`http://www.nist.gov`)

Attacking a block-cipher in this way is known as a *known plaintext-ciphertext* attack, where P_i are the plaintexts and C_i are the ciphertexts. In a realistic attack scenario, whilst the fixed-key may be hidden inside the device used for encryption, the cryptanalyst may be able to either deduce from the context of the message (for instance, an encrypted header of a file), trick or bribe someone to send a specific message — or in a modern context infect a device with malware in order to get the device to send a specific message. Regardless of the method used to obtain the plaintexts, we will wish to examine the resources required to attack this problem via search based methods — specifically quantum search based methods in Chapter 6.

The first problem is that of *plaintext unicity distance*, or the number r of plaintext-ciphertext pairs required to *uniquely* determine the fixed key $K \in \{0,1\}^k$. It will do the cryptanalyst no good if they locate a $K \in \{0,1\}^k$ that satisfies $\vec{P}_r$, but decrypts all other plaintexts to random messages. A good block-cipher will possess the property that if we fix a plaintext $P \in \{0,1\}^n$ then the function $E_P : \{0,1\}^k \longrightarrow \{0,1\}^n$ defined by

$$E_P(x) = E(P, x) \tag{1.31}$$

will act as a pseudorandom function, in that any particular input $x \in \{0,1\}^k$ corresponding to a key has an equal chance of mapping to any element $C \in \{0,1\}^n$ of the co-domain. It is plain that if $k > n$, then some keys must map the fixed plaintext to the same ciphertext. In order to uniquely determine the fixed key with high probability, the cryptanalyst chooses r (in the context of Definition 1.5) to be large enough and specifies the problem as a preimage search problem where $h_r : \{0,1\}^k \longrightarrow \{0,1\}^{rn}$, $y_h = C_1 \| \ldots \| C_r$ and

$$h_r(x) \mapsto E_{P_1}(x) \| \ldots \| E_{P_r}(x). \tag{1.32}$$

In this way we have that the probability of any one key mapping r plaintexts to exactly r ciphertexts is 2^{-rn}. The scenario guarantees that there exists a single key (as we have captured the corresponding plaintext-ciphertext pairs), hence the expected number of solutions to the single-target preimage search problem as defined by h_r and y_h above is $1 + (2^k - 1) \cdot 2^{-rn}$. The expected number of spurious keys is therefore approximately 2^{k-rn}. Hence we will wish to choose r such that this value is as close to 1 as possible, without choosing r to be too large so as to incur additional costs.

For AES, this means that as $n = 128$ is fixed for each each key size $k = 128, 192, 256$, we can choose $r = 2$ for $k = 128, 192$ and $r = 3$ for $k = 256$. This means that we expect 2^{-128} spurious keys for AES-128, an expected 2^{-64} spurious keys for AES-192 and an expected 2^{-128} spurious keys for AES-256. In the original paper on applying Grover's algorithm to cryptanalysis of AES via the preimage search problem, they describe this analysis but choose $r = 3$ for AES-128, $r = 4$ for AES-192 and $r = 5$ for AES-256. This may partly be because the analysis is recycled from a previous paper by two of the authors [RS15] where r is derived based upon the probability that *pairs* of keys encrypt the same plaintext to the same ciphertext, as opposed to our scenario where one key (the key we are attempting to find) is fixed. Their choice of r is therefore not incorrect, but is inefficient for the single-target preimage attack scenario.

We note that classically a filtering process can be used, much as in the case of Fast Exhaustive Search Algorithm described in Section 1.3.4.2 to reduce the dependency of the algorithm on the number of equations. In this scenario, we simply test keys until we find one that fulfils the condition $E_{P_1}(x) = C_1$, then proceed to test the key to check if it satisfies the remaining $r - 1$ plaintext-ciphertext pairs. This is impossible in Grover's algorithm as the nature of implementing boolean circuits in quantum circuitry does not allow conditional execution to work in this fashion. Our adaptation in Chapter 6 is essentially a quantum analogue of this filtering system, building upon the work of [KYYLHH15] and using amplitude amplification to achieve this.

Chapter 2

Quantum computing

> *The first thing to realise about parallel universes, the Guide says, is that they are not parallel. It is also important to realise that they are not, strictly speaking, universes either, but it is easiest if you try and realise that a little later, after you've realised that everything you've realised up to that moment is not true.*
>
> - Douglas Adams, *Mostly Harmless [Ada92]*

In this chapter we provide the required background in quantum computing to understand the quantum search algorithms and cost model contained in this thesis. This is not a thesis about how quantum computers work, but rather on how to optimise and use quantum search algorithms. We will rely upon a simple abstraction of quantum computing which requires only that we accept some basic principles as facts and provide discussion around the subject.

In Section 2.1 we introduce the *mathematical model of quantum computation*, relating briefly as to its connection to the postulates of quantum mechanics. The mathematical model of quantum computing allows us to describe quantum algorithms in the language of linear algebra, with the quantum state being represented by a unit vector in a finite-dimensional complex Hilbert space and quantum algorithms being the space of all unitary operators acting upon this Hilbert space. The mathematical model is a particularly useful tool in regards to the creation of quantum algorithms and the proof of their correctness, as quantum algorithms may be described in terms of unitary operators given only in terms of a high-level description.

In Section 2.2 we cover the *logical quantum circuit model of computation*, which concerns itself with how the abstract unitary operators in the mathematical model of quantum computation may be implemented via a finite set of quantum gates acting on only one or two qubits. A finite set of quantum gates which enable any theoretical quantum algorithm to be implemented, at least up to an arbitrarily chosen level of error, is known as a *universal quantum gate set* (see Section 2.2.1) and we provide the cost of our quantum algorithms in terms of the Clifford+T universal quantum gate set (see Section 2.2.2).

In Section 2.3 we discuss the topic of quantum phase oracles and how they can be constructed via implementing boolean primitives in quantum circuitry. We discuss design patterns for quantum oracles in Section 2.4 and conclude with explicit details of how the quantum phase oracle for the $\mathcal{MQ}$ problem may be implemented in Section 2.5.

Whilst the theory of quantum algorithms and quantum computation is progressing, the eventual architecture(s) that quantum computers will utilise is still an open question. This thesis ignores concerns such as how qubits will be implemented, the internal topology of quantum computers and full details of error-correction. These are important real-world concerns, but ignoring them is both sensible from a technical and sociological perspective. From the technical perspective, the cost of executing a fault-tolerant quantum circuit is extrapolated from the cost of executing a noise-free quantum circuit, hence providing the cost of a quantum algorithm in the noise-free quantum circuit model of computation allows us to crudely compare quantum algorithms, without tying us into any assumptions concerning the engineering principles of any future implementation of a quantum computer.

We will therefore ignore the issue of error-correction in this thesis, as optimisations in the logical circuit-model of computation are both relevant in terms of cryptanalysis and will be expected to carry over to an implementation which involves quantum error correction if the techniques involve strict-gains for all metrics for the logical quantum circuit. The impact of error-correction upon optimisations which involve advantageous tradeoffs relative to metrics in the logical quantum circuit-model of computation is trickier to analyse, as the true impact will depend upon how these metrics impact upon the particular quantum error-correction scheme.

From a sociological perspective, the cryptographic currently uses the noise-free quantum-circuit model, sometimes refered to as the logical quantum circuit model, as a means of comparing quantum algorithms and extrapolating security parameters. Hence, in some respect it does not matter if the quantum algorithms can ever be implemented, or what their true execution costs will be — only that the cryptographic community is currently making informed decisions based upon the estimated noise-free quantum-circuit complexity of these algorithms. Such *quantum resource estimates* have been performed for cryptanalysis of AES [GLRS16] with the Clifford+T universal gate set, the Multivariate Quadratic problem over $\mathbb{F}_2$ using logical quantum gates [SW16], the Elliptic Curve Discrete Logarithm problem using only Toffoli gates [RNSL17] and for solving the SHA preimage problem with Grover's algorithm by using a cost-model based upon the projected classical resources required for a fault-tolerant implementation [ADMG$^+$16]. This study of the resources required to implement quantum algorithms (relative to any cost model) has colloquially become known as the study of *quantum resource estimation*.

The use of logical quantum gates as a metric for security estimates with respect to quantum algorithms was strengthened by the announcement of the NIST[1] standardisation process for quantum-resistant public-key cryptosystems [oST16a]. The NIST call for proposals for this standardisation process [oST16b] required submissions to quantify the estimated "computational resources" required to attack submitted cryptosystems with classical or quantum algorithms in comparison with those required to attack the AES-128/192/256 symmetric key primitives and SHA-256/384/512 hash functions. "Computational resources" is a well-understood term with regards to classical bit operations, but as the underlying hardware architectures of classical and quantum computers are fundamentally different, it is currently difficult to make a straight forward comparison — though efforts have been made [ADMG$^+$16].

[1] National Institute of Standards and Technology

The NIST call for proposals therefore suggests using the logical quantum circuit model for security estimates. The NIST call for proposals states that various metrics may eventually be used and that research into metrics for quantum algorithms is ongoing. For now at least, the logical quantum circuit-model is a standard choice for quantifying the resources required to execute a quantum algorithm and the Clifford+T gate set is useful to consider, owing to its use in the analysis of Grover's algorithm applied to AES [GLRS16] and proposed benefits with regards to quantum error-correction schemes.

2.1 The mathematical model of quantum computing

We now introduce the mathematical model of quantum computation, which can in some sense be considered as level of abstraction at which we design quantum algorithms. Our aim is not to be exhaustive in our description of quantum computation, but to provide a model of quantum computation that is suitable for the purpose of this thesis. Much of the following material is adapted from several well-known resources on quantum computation and the interested reader is directed towards these [NC10].

Data structures are at the core of computer science and in some sense quantum computing can be considered as a method of exploiting a unique data structure, the quantum state, whose creation, manipulation and access is enabled by our understanding of quantum mechanics. As with other data-structures, it is built of relatively simple components (qubits), which are connected in a unique manner to create the data structure (the quantum state), has a means of extracting data from the structure (measurement) and a means of manipulating the data structure itself (application of unitary operators).

2.1.1 A word on dirac notation

In the following and in Chapter 3 we use *Dirac notation* to describe quantum states and operations, which are described by vectors over $\mathbb{C}$ of unit length and unitary matrices. The length of these vectors will be implicit — when we are working with n qubits, the vector will a unit length element of $\mathbb{C}^{2^n}$. In all cases throughout this thesis vectors will be relative to the canonical basis $\mathbf{e}_0, \ldots, \mathbf{e}_{2^n-1}$ where $\mathbf{e}_i \in \mathbb{C}^{2^n}$ is zero apart from the i^{th} component, which is 1. For our purposes, this will be known as the *computational basis*. With this in mind, the following may be useful:

- The *ket* $|\psi\rangle$ is an element of $\mathbb{C}^{2^n}$ whose i^{th} component is indexed by the binary expansion of i, padded on the left with zeroes so that the index is a length n bitstring.
- The *bra* $\langle\psi|$ will represent the conjugate-transpose of $|\psi\rangle$, so that (if we treat vectors as column matrices) $\langle\psi| = |\psi\rangle^\dagger$, where $\dagger$ is the notation for the conjugate-transpose.
- The *inner product* $\langle\varphi|\psi\rangle = |\varphi\rangle^\dagger |\psi\rangle = \sum_{i=0}^{2^n-1} \varphi_i^* \phi_i$.
- The notation $\langle\varphi| A |\phi\rangle$ will denote the inner product $\langle\varphi| (A |\phi\rangle)$, where $A \in \mathbb{C}^{2^n \times 2^n}$.
- The notation $|\varphi\rangle |\phi\rangle$ will be used as shorthand for the tensor product $|\varphi\rangle \otimes |\phi\rangle$.

2.1.2 Quantum states and quantum registers

Together, postulate one and postulate two mathematically describe the basic components of the data structure upon which quantum algorithms act.

Postulate one

The space of configurations for which any isolated physical system may be in can be described by a Hilbert space over $\mathbb{C}$. At any point in time the physical system is completely described by a *state vector* $|\psi\rangle$ of unit length in this *state space* so that $\langle\psi|\psi\rangle = 1$.

For our purposes, the physical system will represent the internal state of the quantum computer and the state space will always be a finite dimensional Hilbert space over $\mathbb{C}$ of dimension 2^n, where n is the number of quantum bits, or *qubits*, in our system. These qubits act as the quantum analogue to classical bits and comprise the basic building blocks of the data structure upon which universal quantum computers process information. In the case $n = 1$, the state space of each qubit is described by the two-dimensional complex vector space $\mathbb{C}^2$. If the basis of this vector space is comprised of the orthonormal unit vectors $\{|0\rangle, |1\rangle\}$, then the qubit may be in any *superposition* (see Definition 2.4) of these basis states

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle , \tag{2.1}$$

with $\alpha_0, \alpha_1 \in \mathbb{C}$ and with the unit vector constraint $\langle\psi|\psi\rangle = 1$ giving us that $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

Definition 2.1 (Computational basis state of a single qubit)**.**
The basis $\{|0\rangle, |1\rangle\}$ for the Hilbert space $\mathbb{C}^2$ representing an individual qubit is called the *computational basis* and an arbitrary element $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ is represented by the vector

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \in \mathbb{C}^2. \tag{2.2}$$

In order to consider systems comprised of more than one qubit, we require postulate two.

Postulate two

The state space of a composite physical system may be described by the tensor product of the individual components of the system. If two separate quantum systems are in the states $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$ are treated as a single system, this composite system is in the state

$$|\psi_1\rangle \otimes |\psi_2\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2. \tag{2.3}$$

Together, the first two postulates provides us with a method of describing the state of the quantum system as a whole.

As we will treat the state of the quantum system as memory upon which operations are formed, in the context of quantum algorithms we will refer to the state of the system as the *quantum register*. It will be useful to consider different logical decompositions of the state space when we analyse quantum algorithms and consider the effect of *entanglement*. As such, we will refer to specific subsystems by named quantum registers. For example, in (2.3), we might refer to the subsystems $\mathcal{H}_1$ and $\mathcal{H}_2$ as the first and second register, or the search-space and ancillae registers. As is standard notation, we will often exclude the tensor sign $\otimes$, so that the state $|\psi_1\rangle \otimes |\psi_2\rangle$ is simply written $|\psi_1\rangle |\psi_2\rangle$ or even $|\psi_1\psi_2\rangle$.

As the basis of a tensor of vector spaces is the set of tensors of all combinations of the basis states for the individual subsystems of the tensored vector space, this provides a canonical description of the state of an arbitrary system via the computational basis.

Definition 2.2 (n-qubit computational basis states)**.**
The set of *computational basis states* of an n-qubit quantum state is the set

$$\{|x_1 \dots x_n\rangle = |x_1\rangle \otimes \cdots \otimes |x_n\rangle \ : \ x_1 \dots x_n \in \{0,1\}^n\}. \tag{2.4}$$

As is standard notation, we will write the state alternatively as $|x_1 x_2 \dots x_n\rangle$ or, when the number of qubits is implicit, by simply the notation $|x\rangle = |x_1 \dots x_n\rangle$. Whilst a tensor product can be used to describe a composite system, after these components have interacted they cannot always be described as a tensor product.

Definition 2.3 (Entanglement)**.**
Given a tensor decomposition of a composite quantum system and state vector belonging to this composite quantum system $|\psi\rangle \in \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_n$, we say that $|\psi\rangle$ is *entangled* if it cannot be written as the tensor product

$$|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle \in \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_n, \qquad \text{with } |\psi_i\rangle \in \mathcal{H}_i. \tag{2.5}$$

We note that entanglement must be defined in relation to a specific tensor decomposition of the Hilbert space representing the composite quantum system. For instance, we could define the state $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$ such that $|\psi_1\rangle \in \mathcal{H}_1$ is entangled with regards to the decomposition $\mathcal{H}_1 = \mathbb{C}^2 \otimes \mathbb{C}^2$, but $|\psi\rangle$ is unentangled with regards to the decomposition $\mathcal{H}_1 \otimes \mathcal{H}_2$.

As a simple example of this and to illustrate the vector and ket notation for systems of more than one qubit, we consider the two-qubit *Bell state* $|\beta_{00}\rangle$, which can be written respectively as

$$|\beta_{00}\rangle = \frac{|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle}{\sqrt{2}} = \frac{|0\rangle |0\rangle + |1\rangle |1\rangle}{\sqrt{2}} = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}. \tag{2.6}$$

Whilst this can always be expressed relative to the computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, the state $|\beta_{00}\rangle$ cannot be written as the tensor product $|\varphi\rangle |\phi\rangle$ for any choice of $|\varphi\rangle$ and $|\phi\rangle$.

Relative to the tensor decomposition $\mathbb{C}^2 \otimes \mathbb{C}^2$, the state $|\beta_{00}\rangle$ is entangled as, if there existed constants such that $|\beta_{00}\rangle = \big(a\,|0\rangle + b\,|1\rangle\big) \otimes \big(c\,|0\rangle + d\,|1\rangle\big)$, then we would have that

$$ac = \frac{1}{\sqrt{2}} \qquad ad = 0 \qquad bc = 0 \qquad bd = \frac{1}{\sqrt{2}}, \tag{2.7}$$

which is clearly a contradiction. This can be compared with an unentangled state

$$\frac{1}{2}\big(\,|00\rangle + |01\rangle + |10\rangle + |11\rangle\,\big) = \frac{1}{\sqrt{2}}\big(\,|0\rangle + |1\rangle\,\big) \otimes \frac{1}{\sqrt{2}}\big(\,|0\rangle + |1\rangle\,\big). \tag{2.8}$$

An arbitrary n-qubit quantum system in relation to the computational basis states can therefore be written as

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x\,|x\rangle = \begin{pmatrix} \alpha_{0^n} \\ \vdots \\ \alpha_{1^n} \end{pmatrix}. \tag{2.9}$$

Whilst the set of states $\{|\mathbf{e}_0\rangle, \ldots, |\mathbf{e}_{2^n-1}\rangle\}$ forms an orthonormal basis (in that $\langle \mathbf{e}_i | \mathbf{e}_j \rangle = \delta_{i,j}$), in general a quantum state may be expressed relative to any orthonormal basis. Whilst we will only be concerned with the computational basis $\{|\mathbf{e}_0\rangle, \ldots, |\mathbf{e}_{2^n-1}\rangle\}$ in this thesis, it is important to mention this in relation to the notion of *superposition*.

Definition 2.4 (Superposition and amplitude)**.**
Given an orthonormal basis for an N-dimensional vector space associated with a quantum system, $\{|\psi_i\rangle\}_{i=0}^{N-1}$, we say that a state vector is in a *superposition* with respect to these basis states if

$$|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i\,|\psi_i\rangle\,, \qquad \text{where } a_i \in \mathbb{C} \tag{2.10}$$

and $|\psi\rangle \neq |\psi_i\rangle$ for any i. The $\alpha_i \in \mathbb{C}$ are referred to as the *amplitudes* of the basis state $|\psi_i\rangle$.

It is worth noting that the concept of superposition and amplitude are only relevant with regards to a given orthonormal basis, as we can always define a new basis with the Gram-Schmidt procedure such that the state $|\psi\rangle$ as in (2.10) is defined to be the first basis vector and the computational basis states are modified accordingly to be the remaining vectors in the basis.

Definition 2.5 (Global and relative phase)**.**
Two quantum states are said to differ by a *global phase* of $e^{i\theta}$ if there exists a $\theta \in (0, 2\pi)$ such that $|\psi'\rangle = e^{i\theta}\,|\psi\rangle$. Two amplitudes α and β are said to differ by a *relative phase* if there exists $\theta \in (0, 2\pi)$ such that $\alpha = e^{i\theta}\beta$. In relation to an orthonormal basis $\{|i\rangle\}_{i=0}^{N-1}$,

$$|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i\,|i\rangle \qquad \text{and} \qquad |\psi'\rangle = \sum_{i=0}^{N-1} \alpha'_i\,|i\rangle \tag{2.11}$$

are said to differ by a *relative phase* if for each basis state $|i\rangle$ it holds that α_i and α'_i differ by a relative phase and that $|\psi\rangle$ and $|\psi'\rangle$ do not differ by a global phase.

Finally, we note that the above may be generalised via considering *qudits* instead of qubits, so that each individual qudit is described by a $d \geq 2$ dimensional complex Hilbert space and the state space associated to the entire quantum system comprised of n qudits is of dimension d^n. In this case, we have that qubits are simply the special case of $d = 2$. As in the case of classical complexity, qudits do not change the fundamental complexity of algorithms, though can have significant impact upon the efficiency of implementing certain circuits and on the performance of error-correction schemes.

2.1.3 Measurement

Whilst postulates one and two deal with how we can mathematically describe a *closed* quantum system, postulate three lays the foundations of the nature of information that can be extracted from it by *observation* and how such interaction affects the quantum state. Such observation is not only required for obtaining the end result of a computation, but is also of utility in creating certain useful subroutines [WR14]. We will only require the case of measurement of a quantum system in relation to the computational basis (see Definition (2.4), whilst postulate three describes the general case of *projective* or *Von Neumann measurement*, which is one formulation of quantum measurement.

Postulate three (Projective Measurement)

For an n-qubit system whose state space $\mathcal{H}$ is spanned by the 2^n computational basis states $\{|x\rangle : x \in \{0,1\}^n\}$, it is possible to perform a *Von Neumann measurement* on the system $\mathcal{H}$ with respect to the computational basis. If the quantum state is described by

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle , \tag{2.12}$$

then measurement outputs $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$ and leaves the system in state $|x\rangle$.

The fact that quantum states are described by unit-length vectors is now clearer, as measurement will result in *some* $x \in \{0,1\}^n$, hence we must have that $\sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1$.

Theorem 2.6 (Global phase is irrelevant)**.**
The global phase of a quantum state cannot be observed by measurement and can therefore be ignored or factored out.

Proof. This can be seen by taking two quantum states which differ by a global phase, so that $|\psi'\rangle = e^{i\theta} |\psi\rangle$. If we consider the probability of measuring an arbitrary $x \in \{0,1\}^n$, then we have that the probability of measuring $|\psi\rangle$ and obtaining $x \in \{0,1\}^n$ is $|\alpha_x|^2$, whilst the probability of measuring $|\psi'\rangle$ and obtaining $x \in \{0,1\}^n$ is $|e^{i\theta}\alpha_x|^2 = |e^{i\theta}| \cdot |\alpha_x| = |\alpha_x|$. Hence the probability of measurement is identical. □

Whilst the subject of quantum measurement can be given a more formal treatment, this is all that we will require from quantum measurement in this thesis.

2.1.4 Unitary evolution

The fourth postulate states how *closed* quantum systems evolve — that is, how quantum states change over time when no measurement is performed. Whilst the fact that the postulate describes *closed* systems may imply that we have no control over the system, it can be shown that by using specific experimental techniques, such as a laser shone on an atom at a certain frequency, we can alter the evolution of an otherwise closed quantum system without interferring with any other aspect of it, or measuring the quantum state. This allows for direct manipulation of the quantum state and, alongside measurement, allows for a complete instruction set of the possible operations we can perform in quantum algorithms.

Postulate four

The evolution of a *closed* quantum system is described by a *unitary transformation*. In other words the state of the system at time t is related to the state of the system at time 0 by a unitary operator U which depends only upon the time t

$$|\psi_t\rangle = U\,|\psi_0\rangle\,. \tag{2.13}$$

Definition 2.7 (Unitary operator)**.**
An operator U is *unitary* if $U^\dagger U = I$, where $U^\dagger = (U^T)^*$ is the conjugate-transpose operation.

This immediately characterises all operations, excluding measurement, that we may perform on the quantum state. It also implies that all operations must be *reversible*, as a unitary transformation always has an inverse. This has a direct impact upon the implementation of any classical subroutine in quantum circuitry, as we must take care to implement these circuits in a reversible manner, so that they implement *permutations* on each computational basis state. As we will see in Section 2.3.1, there are effective methods to realise this.

Whilst there exist models of quantum computation, such as *adiabatic quantum computation* which model quantum computation in continuous time, the *circuit model of quantum computation* (see Section 2.2) we use relies upon discrete unitary transformations referred to as *quantum gates*. This use of discrete operations is purely an abstraction that assists in the modelling and design of quantum computers and the design of algorithms, as each operation we perform will, in reality, be a continuous transformation of the quantum state.

Together with the *principal of deferred measurement* (which loosely states that for any quantum algorithm that involves measurement, we can delay that measurement until the end of the computation at the cost of using more qubits), this implies that any quantum algorithm can theoretically be viewed as consisting of three steps:

1. An initialisation phase.
2. A series of unitary transformations upon the state space.
3. Measurement.

This is not to say that intermediate measurement is not useful — it can result in advantageous tradeoffs with regards to the resources used to implement quantum algorithms, in implementations of primitives [WR14] and in quantum error-correction [NC10, FMMC12], but that it is not required for quantum algorithms to be correct. The quantum algorithms we describe in this thesis will not use intermediate measurement and the author is not confident that it can be used to improve any of the methods in this thesis.

It is important to note that postulate four makes no claims concerning the efficiency of implementing unitary operations and a key aspect of the design of quantum algorithms is ensuring that we can efficiently construct useful unitary transformations. Whilst there are an uncountable number of unitary transformations over the state-space of a quantum system, a set of finite unitary operators known as a *universal quantum gate set* (see Section 2.2.1) acting only upon single qubits and pairs of qubits will suffice to emulate any unitary transformation up to an arbitrary level of precision in the quantum circuit model.

2.1.5 Describing unitary transformations

An alternative way of representing unitary transformations is via their outer product representation. For any two vectors $|v\rangle, |w\rangle \in \mathcal{H}$, the *outer product* of these two vectors is defined as the linear operator $|v\rangle\langle w| : \mathcal{H} \longrightarrow \mathcal{H}$ whose action is

$$(|v\rangle\langle w|)|x\rangle \mapsto \langle w|x\rangle |v\rangle . \tag{2.14}$$

This allows us to represent unitary operators via their *outer product representation*, which will be of utility in the proof of Theorem 3.4, a component of the proof of amplitude amplification (see Theorem 3.6).

Theorem 2.8 (Outer product representation of a unitary operator (see Theorem 2.3.2 [KLM07]))**.** Let $B = \{|b_i\rangle\}$ be an orthonormal basis for a vector space $\mathcal{H}$. Then every linear operator U on $\mathcal{H}$ can be written as

$$U = \sum_{b_i, b_j \in B} U_{i,j} |b_i\rangle\langle b_j| \tag{2.15}$$

We will give several examples of this decomposition for illustration in Section 2.1.6.

2.1.6 A collection of basic unitary transformations

We now describe a set of unitary transformations, which will be used throughout this thesis. Just as a composite system can be described by a tensor product, it will be useful to describe a tensor product of unitary operators acting upon the state space. Hence if we have the system $\mathcal{H}_1 \otimes \mathcal{H}_2$, with U_1 acting upon $\mathcal{H}_1$ and U_2 acting upon $\mathcal{H}_2$, then we denote the parallel application of these unitary operators upon their respective vector spaces by the tensor product of the two operators $U_1 \otimes U_2 : \mathcal{H}_1 \otimes \mathcal{H}_2 \longrightarrow \mathcal{H}_1 \otimes \mathcal{H}_2$. When we wish to signify that we are applying the same unitary U to each component of the composite system $\mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_k$, we will use the notation $U^{\otimes k}$.

We now proceed to give the unitary operators for several of the important quantum gates that we will use, to illustrate the above concepts.

2.1.6.1 The Identity operator

The identity unitary operator is the simplest unitary operator that we consider, but is technically required if we are to write other unitary operators as a tensor product of unitary operators, if we are in essence as acting upon parts of the composite system. It has the representation in matrix and outer product form

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = |0\rangle\langle 0| + |1\rangle\langle 1| . \tag{2.16}$$

If we wish to signify that the identity is acting upon the state space $\mathbb{C}^{2^n}$ representing n qubits, we will use either the notation $I^{\otimes n}$ or I_n. The extension of (2.16) to these cases is trivial.

2.1.6.2 The X operator

The X operator has the representation

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = |1\rangle\langle 0| + |0\rangle\langle 1| . \tag{2.17}$$

2.1.6.3 The Hadamard unitary

The Hadamard operator acts upon the state space $\mathbb{C}^2$ corresponding to a single qubit and has the representation

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{\sqrt{2}} |0\rangle \langle 0| + \frac{1}{\sqrt{2}} |1\rangle \langle 0| + \frac{1}{\sqrt{2}} |0\rangle \langle 1| - \frac{1}{\sqrt{2}} |1\rangle \langle 1| . \tag{2.18}$$

2.1.6.4 The Controlled-NOT (Controlled-X, CNOT or $\wedge_1(X)$ unitary

The Controlled-NOT or $\wedge_1(X)$ operator acts upon two qubits, hence acts upon the space $\mathbb{C}^4$ and has the representation

$$\wedge_1(X) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = |00\rangle \langle 00| + |01\rangle \langle 01| + |11\rangle \langle 10| + |10\rangle \langle 11| . \tag{2.19}$$

The actual unitary representation of the algorithms we use will not be important for our purposes, other than the explicit description of the Hadamard operator given above. The outer product representation in relation to a given orthonormal basis on the other hand, will prove useful. We now summarise the key take-home points concerning the mathematical model of quantum computation and move on to discuss the quantum circuit model of computation.

2.1.7 A summary of the mathematical model of quantum computation

We briefly recap the previous section, noting the key points that we must keep in mind.

- *Quantum states* consisting of n-qubits may be considered to be unit-length complex vectors dimension 2^n. The set of all possible n-qubit states is the set of unit-length complex vectors of dimension 2^n. In general, a quantum state may be represented as

$$|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i |\psi_i\rangle , \tag{2.20}$$

  where $\{|\psi_i\rangle\}_{i=0}^{2^n-1}$ is any orthonormal basis for the vector space $\mathcal{H} = \mathbb{C}^{2^n}$ that represents the state space $\mathcal{H}$ of possible configurations for the quantum state and the unit-length condition that $\sum_{i=0}^{N-1} |\alpha_i|^2 = 1$ must hold.

  Relative to the computational basis $\{|x\rangle : x \in \{0,1\}^n\}$, the components of these vectors are labelled α_x, so that the i^{th} component of this vector is the amplitude associated with $|x\rangle$, where x is the binary interpretation of i (where $i = 0, \ldots, 2^n - 1$).

- *Quantum algorithms* act upon n-qubit quantum states and may be considered to be unitary operators (see Definition 2.7). The space of all possible quantum algorithms (involving no measurement) that act upon n-qubit states is therefore described by the set of all possible unitary operators of dimension $2^n \times 2^n$ over $\mathbb{C}$, ie. $\left\{U \in \mathbb{C}^{2^n \times 2^n} : U^\dagger U = U U^\dagger = I_{2^n}\right\}$.

- *Quantum measurement* of the entire quantum state (all we will require in this thesis) allows us to collapse the n-qubit state into one of 2^n computational basis states, which provides us with a method of extracting classical information from the quantum state. The probability of measuring the state

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \tag{2.21}$$

  and obtaining the bitstring x is $|\alpha_x|^2$. Measurement will only be performed at the end of a quantum computation in this thesis.

Whilst the mathematical model of quantum computation is a useful tool for algorithm design (see Chapter 3), the quantum circuit model of computation is a useful tool for deriving the costs of these algorithms.

2.2 The quantum circuit model of computation

Whilst the mathematical model of quantum computation gives us the basic data structure of the quantum state and tells us how we can extract information via measurement from it, a quantum algorithm is simply described as a unitary operator acting upon the state space. The quantum circuit model of computation helps quantify the resources required to implement these unitary operators via operations on a small number of qubits, much as classical circuits can be considered as functions which can be implemented via logical gates which act upon a small number of bits. For our purposes, a *quantum circuit* will consist of

- A finite number of *quantum wires*, each of which represent a qubit.
- An initialisation phase where all qubits are set to $|0\rangle$. This can be performed via measuring any quantum state and simply applying an X (see Section 2.1.6.2) gate to qubits measured in the state $|1\rangle$.
- The application of *quantum gates* to subsets of these wires.
- Measurement of the quantum state at the end of the computation.

Other features, such as intermediate measurement of the state and conditional application of quantum gates based upon these measurements are required by the full quantum circuit model of computation, but are not required for our purposes as we only perform measurement during the initialisation phase and at the end of the quantum computation.

A *quantum gate* is loosely defined as the implementation of a unitary operator that acts upon a small number of qubits. It is important that these gates are fixed in advance in order to quantify the resources required by the quantum algorithm. Some authors include all possible single-qubit gates and the Toffoli gate [AdW17], whilst others fix a finite set of quantum gates [GLRS16] known as a *universal quantum gate set* (see discussion on the next page) such as the Clifford+T gate set (see Section 2.2.2), which consists of a finite number of quantum gates that can be used to approximate any given single-qubit unitary transformation. The question of approximating gates will not be important for the majority of this thesis, as our algorithms can be implemented exactly using our choice of the Clifford+T universal quantum gate set. We mention the arbitrary approximation of single-qubit unitaries as one of the algorithms that we compare our methods to in Chapter 6 requires this.

The quantum circuit model is similar to the classical circuit model, in that the metrics we care about are

- Circuit-width — the number of qubits used in a quantum circuit.
- Circuit-depth — the number of time-steps we must perform before the computation ends, where quantum gates may be executed in parallel in each time-step.
- Circuit-size — the total number of primitive quantum gates used in the quantum circuit (though individual gates may possess different real-world costs).

In particular, it is the quantum circuit-size of quantum search algorithms that we are interested in, as the quantum security levels of Gui and AES are benchmarked against these metrics. Quantum-circuit depth is considered less important in the cryptographic community when it comes to applying brute-force search methods, but quantum circuit-width has a very real impact upon the timeline for if or when schemes become vulnerable to attack from quantum computers. Considering the fact that we currently have quantum computers consisting of just under one hundred *physical* qubits and that quantum search algorithms will require *logical* qubits, each consisting of thousands of physical qubits, circuit-width is a very real-world concern. Our results in Chapter 6 are aimed towards this, demonstrating that we can achieve better results with regards to circuit-depth and circuit-size than Grover's algorithm (see Theorem 3.10) can achieve if we use a finer-grained approach via exploiting amplitude amplification (see Theorem 3.6).

It can be useful to visualise a quantum circuit via a *quantum circuit diagram*, which for a quantum circuit acting upon n qubits, consists of n parallel lines or *quantum wires*, representing the qubits. The individual quantum gates or quantum algorithms can then be represented via boxes or control-lines acting upon these wires. Figure (2-1) below illustrates a quantum circuit that maps the state $|xy\rangle$ (where $x, y \in \{0, 1\}$) to the *Bell state* $|\beta_{xy}\rangle$, which is a general circuit which can create the Bell state $|\beta_{00}\rangle$ as given in (2.6). The circuit consists of first applying a Hadamard gate (the implementation of a Hadamard unitary) to the first qubit, then applying the $\wedge_1(X)$ gate (the implementation of the two qubit $\wedge_1(X)$ or controlled-NOT gate) to both qubits, with the first qubit being the *control qubit* and the second qubit being the *target qubit*. The $\wedge_1(X)$ gate and its generalisation $\wedge_k(X)$ is further discussed in Section 2.3.1.

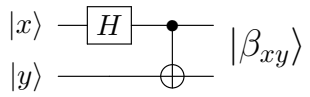


Figure 2-1: The Bell state quantum circuit, mapping the computational basis $|xy\rangle$ to $|\beta_{xy}\rangle$.

In terms of unitary operations, this should be interpreted as first applying the tensor of the Hadamard unitary with the identity $(H \otimes I)$, followed by the $\wedge_1(X)$ unitary. Written out explicitly, this circuit applied to the state $|00\rangle$ gives us

$$(\wedge_1(X))\,(H \otimes I)\,|00\rangle = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.22}$$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} \tag{2.23}$$

$$= \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle = |\beta_{00}\rangle\,. \tag{2.24}$$

2.2.1 Universal quantum gate sets

Much as a classical circuit may be implemented by a universal logic gate (or set of gates), such as the NAND gate (or the set $\{\neg, \oplus, \wedge\}$), we will discuss the implementation costs of quantum algorithms relative to a fixed set of finite gates. As postulate four (see Section 2.1.4) states that the space of all possible quantum operations on a single qubit is the set of unitary operations $\{U \in \mathbb{C}^{2\times 2}\}$, no finite sequence from a finite set of gates can implement these exactly.

When it comes to implementing these arbitrary single-qubit unitary transformations as single-qubit quantum gates, it can be demonstrated by the Solovay-Kitaev Theorem [KIT96] that a finite set of quantum gates acting upon single qubits can suffices to implement any given single-qubit unitary transformation up to an arbitrarily chosen level of precision and that furthermore, this is efficient in the sense that we require only $O(\log_2^c(\frac{1}{\epsilon}))$, where $\epsilon > 0$ is the chosen level of precision and $1 < c < 4$. As it can be demonstrated that the error in implementing the unitary

$$U_\epsilon = U_{\epsilon_k} \cdots U_{\epsilon_1}, \tag{2.25}$$

where the precision error of implementing U_{ϵ_i} is ϵ_i, is bounded by the sum of the errors $\sum_{i=1}^{k} \epsilon_i$, this gives an easy method of choosing the level of precision we must implement any given unitary to achieve a given tolerance level of error for the quantum algorithm as a whole. In order to implement any quantum algorithm up to a level of precision $\epsilon > 0$, if we are implementing m quantum gates approximately, then we must implement each quantum gate up to a precision level of $\epsilon_i \leq \frac{\epsilon}{m}$ [NC10]. The Solovay-Kitaev theorem implies that if our algorithm uses m single-qubit quantum gate that we are approximating, then we expect to execute $O(m \log^c(\frac{m}{\epsilon}))$ gates from the universal quantum gate set. Asymptotically optimal ($c = 1$) algorithms exist [KBRY15, RS14], but our results rely upon the existence, not the efficiency of arbitrary single-qubit quantum gates.

It can be demonstrated that arbitrary unitary operators acting upon any number of qubits can be implemented via a $\wedge_1(X)$ gate (which acts upon pairs of qubits) in combination with arbitrary single-qubit unitary operators (see [NC10] Section 4.5.2). A finite set of gates which can implement any unitary transformation on n qubits is known as a *universal quantum gate set* and by the discussion above, a method to implement arbitrary single-qubit gates up to a given level of precision combined with the $\wedge_1(X)$ gate is a universal quantum gate set.

For our purposes it is the correctness of methods involving these gates, rather than the costs involved that we are concerned with and it suffices that they can be implemented with only a small overhead as they will only be needed in Theorem 3.11, which we reference rather than use. The algorithms in this thesis can all be implemented precisely using the Clifford+T quantum gate set, which is our fixed choice of universal quantum gate set. We specifically chose this gate set owing to the fact that we are discussing concrete costs, that this gate set is a popular choice in literature owing to its proposed utility in implementing fault-tolerant quantum computation [FMMC12] and that it has previously been used to cost the resources required for the quantum oracle to solve the key-search problem for the Advanced Encryption Standard [GLRS16] (see Section 2.6). We now examine the quantum gates that we will count as primitive operations.

2.2.2 The Clifford+T universal gate set

We count the set of gates

$$\text{Clifford+T} := \{H, S, \wedge_1(X)\} \cup \{T\} \tag{2.26}$$

as the Clifford+T universal quantum gate set and additionally include the X and Z gates as quantum gate primitives. We note that the the H, S and T gates all act upon single qubits, whilst the $\wedge_1(X)$ gate (the controlled-X/controlled-NOT/CNOT gate) acts upon two qubits. It will not prove necessary to discuss the action of the S (phase) gate or the T gate as we do not use them directly — these gates will only be a component of the cost of the primitives described in Section 2.3.1, which describes how boolean functions can be implemented using the Clifford+T quantum gate set. Nevertheless, we include their unitary transformations for completeness. We count each of these gates as having a cost of 1, though will perform a gate count in certain areas of this thesis in terms of both the Clifford+T quantum gates and separately for T gates.

2.2.3 Implementing the T gate in fault-tolerant quantum computation

The circuit-complexity of a quantum circuit is often separated into that required for gates from the Clifford gate set $\{H, S, \wedge_1(X)\}$ and the circuit-size and circuit-depth for the T gate on its own. This owes itself to the difficulty and cost of implementing T-gates in a fault-tolerant fashion. Fault-tolerant quantum computation is a topic beyond the scope of this thesis, but we sketch the arguments for this separation of costs. Whilst the T gate has the unitary representation

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \tag{2.27}$$

the fault-tolerant implementation [FMMC12, ADMG$^+$16, NC10] of this unitary operation is performed via *state injection* of the logical quantum state

$$|\Theta\rangle := \frac{|0\rangle + e^{i\pi/4}|1\rangle}{\sqrt{2}}, \tag{2.28}$$

which is consumed upon the application of each T gate by the process of *state injection*, which consists of performing the application of several gates based upon the outcome of a measurement. Constructing the state $|\Theta\rangle$ to the required level of precision is a costly process, referred to as *magic state distillation* which must be performed to an error tolerance that scales with the inverse of the number of T gates in the entire quantum circuit. Each T gate therefore takes magnitudes more effort to implement than any of the those from the Clifford gate group $\{H, S, \wedge_1(X)\}$ and hence the costs are often given separately, with T count and depth being optimisation targets.

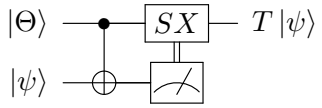
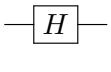


Figure 2-2: State injection circuit for implementation of the T gate (see [NC10] pp.486).

2.2.4 Our set of primitive Clifford+T quantum gates

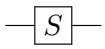
We note that the X and Z gates are not technically part of the Clifford+T gate set, but include them as a primitive gate as they are both relatively simple gates and we have the identities $X = HSSH$ and $Z = S^2$ if we examine the unitary transformations below. Each gate below will be considered to take a single time-step to execute.

The H (Hadamard) gate



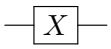
$$\equiv \quad \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.29}$$

The S (Phase) gate



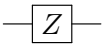
$$\equiv \quad \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{2.30}$$

The X gate



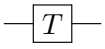
$$\equiv \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.31}$$

The Z gate



$$\equiv \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.32}$$

The T gate



$$\equiv \quad \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \tag{2.33}$$

The $\wedge_1(X)$ gate

$$\equiv \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.34}$$

We will return to the H gate in Section 2.3, which deals with how *quantum oracles* may be constructed and the X and $\wedge_1(X)$ gates in Section 2.3.1, where they will be used to implement boolean primitives in quantum circuitry.

2.2.5 The phase-shift gate

In addition to these gates, it will be useful to define the *phase-shift gate* R_ϕ. The R_ϕ gate is not part of the Clifford+T gate set, but is a gate which (apart from special cases) must be synthesised using the methods discussed in Section 2.2.1.

The R_ϕ gate (where $0 \leq \phi < 2\pi$)

$$-\boxed{R_\phi}- \quad \equiv \quad \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \tag{2.35}$$

2.2.6 A summary of the quantum circuit-model of computation

We have discussed the logical quantum circuit-model of computation and have a primitive set of quantum gates that will be the basis for costing the quantum algorithms designed in the mathematical model of quantum computation. We will see these primitives used in Section 2.3.1, where we construct quantum circuits which will be used in the amplitude amplification subroutine discussed in Chapter 3. The important points to take home are:

- Quantum gates are simply the implementation of unitary operators acting upon a small number of qubits.
- Any given unitary operator acting upon a quantum state vector can be implemented up to an arbitrary level of precision via a universal quantum gate set.
- Our chosen universal quantum gate set is the Clifford+T gate set.
- The important metrics are the number of logical qubits, circuit-size and circuit-depth.

2.3 Quantum oracles and evaluations

Quantum phase oracles can be considered the quantum equivalent of a classical circuit which computes a boolean function which decides whether $x \in \{0,1\}^n$ possesses a particular property. Whereas the output of a classical circuit would be a single bit which indicates whether or not the element x satisfies some condition, the quantum phase oracle performs a conditional change in the phase associated to each computational basis state which possesses this property. In terms of the unstructured search problem (see Definition 1.1), this property is simply defined as being a solution to our search problem, but it will be useful to define property testing of length n bitstrings by considering *boolean indicator functions* later in this thesis.

Definition 2.9 (Boolean indicator function)**.**
We say that the function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ is a *boolean indicator function* for the set of bitstrings $x \in \{0,1\}^n$ in that it partitions the set into two disjoint subsets $\chi^{-1}(1)$ and $\chi^{-1}(0)$.

A classical circuit for $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ therefore takes as input $x \in \{0,1\}^n$ and outputs $\chi(x) = 1$ if x satisfies the property we are looking for and outputs $\chi(x) = 0$ otherwise. In contrast, amplitude amplification exploits *quantum phase oracles*, which manipulate the *phase* of the amplitudes $\{\alpha_x : x \in \{0,1\}^n\}$ associated with the set of computational basis states $\{|x\rangle : x \in \{0,1\}^n \wedge \chi(x) = 1\}$.

Definition 2.10 (Quantum phase oracle defined by a boolean indicator function)**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $0 \leq \phi < 2\pi$. We define the *quantum phase oracle* $\mathcal{O}_\chi(\phi)$ to be the unitary operator acting upon the n qubit computational basis state $|x\rangle$, where $x \in \{0,1\}^n$

$$\mathcal{O}_\chi(\phi)\,|x\rangle \mapsto \begin{cases} e^{i\phi}\,|x\rangle & \text{if } \chi(x) = 1 \\ |x\rangle & \text{if } \chi(x) = 0. \end{cases} \tag{2.36}$$

When $\phi = \pi$, so that $\mathcal{O}_\chi\,|x\rangle \mapsto (-1)^{\chi(x)}\,|x\rangle$, we will simply use the notation $\mathcal{O}_\chi$.

We will use the definition of the generalised quantum phase oracle within the proof of amplitude amplification (see Theorem 3.6) and note that it is one of several ways that *exact amplitude amplification* (see Theorem 3.11) may be implemented [BHMT02], but we will exploit the case where $\phi = \pi$ for the majority of results in this thesis.

The quantum phase oracle therefore does not change the probability of measuring the state and obtaining any particular bitstring $x \in \{0,1\}^n$ on its own, as it only affects the relative phase of the amplitudes of each individual computational basis states and not the magnitude. Amplitude amplification (see Theorem 3.6) will later exploit this so-called *phase-kickback* conditioned on a boolean function in conjunction with additional unitary operators in order to manipulate the magnitude of the amplitudes associated to each computational basis state.

Quantum phase oracles themselves can be implemented in a variety of manners, the simplest of which is via *quantum bit oracles*, which act act upon $n+m$ qubit states and compute the value of an arbitrary boolean vector-valued function $h : \{0,1\}^n \longrightarrow \{0,1\}^m$ for each computational basis state $|x\rangle$, but do not alter the phase.

Definition 2.11 (Quantum bit oracle)**.**
Let $h : \{0,1\}^n \longrightarrow \{0,1\}^m$. We define the *quantum bit oracle* $\mathcal{O}_h^{(b)}$ to be the unitary operator acting upon the $n+m$ qubit state $|x\rangle |b\rangle$, where $x \in \{0,1\}^n$ and $|b\rangle$ is any m qubit state where

$$\mathcal{O}_h^{(b)} |x\rangle |b\rangle \mapsto |x\rangle |b \oplus h(x)\rangle . \tag{2.37}$$

Lemma 2.12 (Implementation of the quantum phase oracle $\mathcal{O}_\chi$ via quantum bit oracles [NC10])**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $|-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$. The action of the quantum phase oracle $\mathcal{O}_\chi$ on the n qubit state $|x\rangle$, where $x \in \{0,1\}^n$ may be implemented by a single application of the quantum bit oracle $\mathcal{O}_\chi^{(b)}$ upon the $n+1$ qubit state $|x\rangle |-\rangle$, so that

$$(\mathcal{O}_\chi \otimes \mathcal{I}) |x\rangle |-\rangle = \mathcal{O}_\chi^{(b)} |x\rangle |-\rangle . \tag{2.38}$$

If a single qubit is initialised and kept in the state $|-\rangle = HX |0\rangle$, then the computational resources required to implement $\mathcal{O}_\chi$ are then identical to those required to implement $\mathcal{O}_\chi^{(b)}$.

Proof. This can be seen via direct computation as

$$\mathcal{O}_\chi^{(b)} |x\rangle |-\rangle = |x\rangle |- \oplus \chi(x)\rangle = (-1)^{\chi(x)} |x\rangle |-\rangle = (\mathcal{O}_\chi \otimes \mathcal{I}) |x\rangle |-\rangle \tag{2.39}$$

as

$$|- \oplus 0\rangle = \frac{|0 \oplus 0\rangle - |1 \oplus 0\rangle}{\sqrt{2}} = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = \quad |-\rangle \tag{2.40}$$

and

$$|- \oplus 1\rangle = \frac{|0 \oplus 1\rangle - |1 \oplus 1\rangle}{\sqrt{2}} = \frac{|1\rangle - |0\rangle}{\sqrt{2}} = - |-\rangle . \tag{2.41}$$

□

The quantum state $|-\rangle$ can be easily initialised at the beginning of the quantum computation via the application of an X gate to the $|0\rangle$ state followed by a Hadamard (H) gate, so that

$$HX |0\rangle = H |1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle . \tag{2.42}$$

An alternative method of implementing $\mathcal{O}_\chi$ via $\mathcal{O}_\chi^{(b)}$ is to use an ancillae initialised to $|0\rangle$, two applications of $\mathcal{O}_\chi^{(b)}$ and a single Z gate each time we wish to compute $\mathcal{O}_\chi$ so that we have

$$\mathcal{O}_\chi^{(b)\dagger} \left(\mathcal{I}^{\otimes n} \otimes Z\right) \mathcal{O}_\chi^{(b)} |x\rangle |0\rangle = \mathcal{O}_\chi^{(b)\dagger} \left(\mathcal{I}^{\otimes n} \otimes Z\right) |x\rangle |\chi(x)\rangle = (-1)^{\chi(x)} \mathcal{O}_\chi^{(b)\dagger} |x\rangle |\chi(x)\rangle = (-1)^{\chi(x)} |x\rangle |0\rangle . \tag{2.43}$$

At first glance this appears to be almost twice as expensive as the method described in Lemma 2.12, but neither decomposition takes into account the fact that implementing quantum bit oracles for non-trivial boolean indicactor functions usually requires ancilla qubits to implement and *quantum evaluations*, which must be executed once to compute $\chi(x)$ and then run in reverse to ensure that the ancillae qubits used form a tensor product with the rest of the quantum state.

Definition 2.13 (Quantum evaluation)**.**
Let $h : \{0,1\}^n \longrightarrow \{0,1\}^m$. We define the *quantum evaluation* $\mathcal{E}_h$ to be the unitary operator acting upon the $w+m+n$ qubit state $|0^w\rangle |0^m\rangle |x\rangle$, where $x \in \{0,1\}^m$

$$\mathcal{E}_h |0^w\rangle |0^m\rangle |x\rangle \mapsto |g(x)\rangle |h(x)\rangle |x\rangle \tag{2.44}$$

and $g(x) \in \{0,1\}^w$ is the end state of the working memory used to compute $h(x) \in \{0,1\}^m$.

Lemma 2.14 (Implementation of quantum bit oracles via quantum evaluations)**.**
Let $h : \{0,1\}^n \longrightarrow \{0,1\}^m$. The action of the quantum bit oracle $\mathcal{O}_h^{(b)}$ on the $n+m$ qubit state $|x\rangle |b\rangle$, where $x \in \{0,1\}^n$ and $|b\rangle$ is any m qubit state can be implemented by one application of $\mathcal{E}_h$, one application of $\mathcal{E}_h^\dagger$ and m $\wedge_1(X)$ gates, so that

$$\left(\mathcal{I}^{w+m} \otimes \mathcal{O}_h^{(b)}\right) |0^w\rangle |0^m\rangle |x\rangle |b\rangle = |0^w\rangle |0^m\rangle |x\rangle |b \oplus h(x)\rangle \tag{2.45}$$

is equivalent to the compute-copy-uncompute paradigm given by the sequence

$$\begin{aligned} |0^w\rangle |0^m\rangle |x\rangle |b\rangle &\overset{\mathcal{E}_h}{\mapsto} |g(x)\rangle |h(x)\rangle |x\rangle |b\rangle \\ &\overset{\wedge_k(X)^{\otimes m}}{\mapsto} |g(x)\rangle |h(x)\rangle |x\rangle |b \oplus h(x)\rangle \\ &\overset{\mathcal{E}_h^\dagger}{\mapsto} |0^w\rangle |0^m\rangle |x\rangle |b \oplus h(x)\rangle . \end{aligned} \tag{2.46}$$

Proof. By inspection and equivalence of (2.45) and (2.46). □

By lemmas 2.12 and 2.14, we can therefore implement the quantum phase oracle $\mathcal{O}_\chi$ for any $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ so long as we can implement the quantum evaluation $\mathcal{E}_\chi$.

Lemma 2.15 (Implementation of the generalised quantum phase oracle)**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $0 \leq \phi < 2\pi$. The generalised quantum phase oracle $\mathcal{O}_\chi(\phi)$ can be implemented via using $n+w+1$ qubits and the sequence $\mathcal{E}_\chi^\dagger \left(\mathcal{I}^{\otimes n+w} \otimes R_\phi\right) \mathcal{E}_\chi$.

Proof. We recall that the single-qubit phase-shift gate (see Section 2.2.5) is defined by the unitary mapping $R_\phi |x\rangle \mapsto (e^{i\phi})^x |x\rangle$, where $x \in \{0,1\}$. We therefore have that

$$\mathcal{E}_\chi^\dagger \left(\mathcal{I}^{\otimes n+w} \otimes R_\phi\right) \mathcal{E}_\chi |0^w\rangle |x\rangle |0\rangle \mapsto \mathcal{E}_\chi^\dagger \left(\mathcal{I}^{\otimes n+w} \otimes R_\phi\right) |g(x)\rangle |x\rangle |\chi(x)\rangle \tag{2.47}$$

$$\mapsto \mathcal{E}_\chi^\dagger (e^{i\phi})^{\chi(x)} |g(x)\rangle |x\rangle |\chi(x)\rangle \tag{2.48}$$

$$\mapsto (e^{i\phi})^{\chi(x)} |0^w\rangle |x\rangle |0\rangle \tag{2.49}$$

□

We therefore have several methods of implementing quantum phase oracles — an ancilla-based method exploiting the state $|-\rangle$ for the case where $\phi = \pi$ and a method for the generalised case.We now turn how we can concretely implement quantum evaluations via quantum circuitry.

2.3.1 Contructing quantum evaluations for arbitrary boolean functions

We rely upon on the well-known fact that the set of boolean logic gates $\{\wedge, \oplus, \neg\}$ is a universal gate set, in that they can be used to create a circuit which computes any given boolean-vector-valued function $h : \{0,1\}^n \longrightarrow \{0,1\}^m$. The exact unitary operations we use are the set $\{X, \wedge_1(X), \wedge_2(X)\}$ — which are also known respectively as the X gate, the controlled-NOT (or CNOT) gate and the Toffoli gate. Other gates will prove are useful, such as the $\wedge_k(X)$ gate for $k > 2$ (sometimes referred to as the k-fold controlled-NOT gate, $k+1$ bit Toffoli gate or $k+1$-bit Multiple Control Toffoli (MCT) gate), $SWAP$ and $\wedge_1(SWAP)$ (the Fredkin) gates, but can be implemented directly via the set $\{X, \wedge_1(X), \wedge_2(X)\}$. We now list these quantum gates and their implementation costs in the Clifford+T universal quantum gate set (see Section 2.2.2), providing their notation in the quantum circuit-model of computation. We list their costs in the Clifford+T universal gate set in Table 2.1.

2.3.1.1 The X gate

The X gate acts upon the single qubit basis state $|x\rangle$, where $x \in \{0,1\}$ and can be interpreted as flipping this bit or performing the addition $x \oplus 1$, so that

$$X|x\rangle \mapsto |\neg x\rangle \qquad \text{or equivalently} \qquad X|x\rangle \mapsto |x \oplus 1\rangle . \tag{2.50}$$

X has the quantum circuit representation

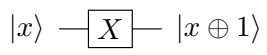


Figure 2-3: The X gate.

We will treat the X gate as a primitive Clifford gate.

2.3.1.2 The $\wedge_1(X)$ or controlled-NOT (CNOT) gate

The $\wedge_1(X)$ gate acts upon the two qubit basis state $|x_1\rangle|x_2\rangle$, where $x_i \in \{0,1\}$, and can be interpreted as performing the reversible exclusive-or operation $x_2 \oplus x_1$, so that

$$\wedge_1(X)|x_1\rangle|x_2\rangle \mapsto |x_1\rangle|x_1 \oplus x_2\rangle . \tag{2.51}$$

The first qubit is known as the *control qubit* whilst the second is known as the *target qubit*. $\wedge_1(X)$ has the quantum circuit representation

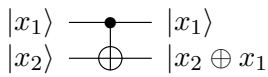


Figure 2-4: The $\wedge_1(X)$ gate.

The $\wedge_1(X)$ is also known as the controlled-NOT or CNOT gate and is a primitive Clifford gate.

2.3.1.3 The $\wedge_2(X)$ or Toffoli gate

The $\wedge_2(X)$ (or *Toffoli*) gate acts upon the three qubit basis state $|x_1x_2\rangle\,|x_3\rangle$, where $x_i \in \{0,1\}$, and can be interpreted as performing the reversible multiplication $x_1 \wedge x_2 = x_1 \cdot x_2$, so that

$$\wedge_2(X)\,|x_1x_2\rangle\,|x_3\rangle \mapsto |x_1x_2\rangle\,|x_1 \oplus (x_1 \wedge x_2)\rangle\,. \tag{2.52}$$

The qubits x_1 and x_2 are known as the *control qubits* whilst x_3 is known as the *target qubit*. $\wedge_2(X)$ has the quantum circuit representation

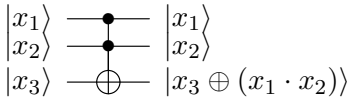


Figure 2-5: The $\wedge_2(X)$ gate.

The Toffoli gate can be implemented in a number of ways, but it has been proven that it requires a minimum of 7 T gates to implement [GKMR14]. It can be implemented efficiently with a T-depth of 4 using no additional ancillae, or with a T-depth of 1 with four clean ancillae [AMM14]. We will use the following quantum circuit for the Toffoli gate for our quantum resource estimations

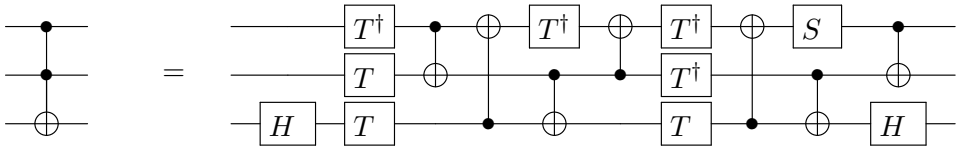


Figure 2-6: The logical Toffoli gate decomposed into Clifford+T gates [AMMR13, Sel13].

which has a Clifford count of 10, a T-count of 7, a T-depth of 3 and an overall depth of 10.

2.3.2 The $\wedge_k(X)$ gate (for $k \geq 3$)

The $\wedge_k(X)$ (sometimes referred to as the k-fold controlled-NOT gate, the $k+1$-bit Toffoli gate or the Multiple Control Toffoli (MCT) gate), acts upon the $k+1$ qubit basis state $|x_1 \ldots x_k\rangle\,|x_{k+1}\rangle$, where $x_i \in \{0,1\}$, and is a generalisation of the $\wedge_1(X)$ and $\wedge_2(X)$ gates so that

$$\wedge_k(X)\,|x_1 \ldots x_k\rangle\,|x_{k+1}\rangle \mapsto |x_1 \ldots x_k\rangle\,|x_{k+1} \oplus (x_1 \wedge \cdots \wedge x_k)\rangle\,. \tag{2.53}$$

The qubits $x_1, \ldots, x_k$ are referred to as the *control qubits* whilst x_{k+1} is referred to as the *target qubit*, with $\wedge_k(X)$ possessing the quantum circuit- representation

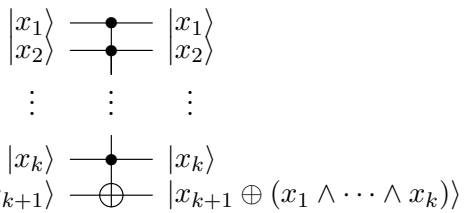


Figure 2-7: The $\wedge_k(X)$ gate.

There are a variety of methods to implementing the $\wedge_k(X)$ gate, the most basic of which is via a network of Toffoli gates [BBC$^+$95] using one ancilla qubit which begins and ends in the state $|0\rangle$. Other methods exploit *relative phase Toffoli gates* in order to make computational gains over this method [Mas16]. Both of these methods use at least one ancilla qubit and $O(k)$ quantum gates and it is known that implementing the $\wedge_k(X)$ gate will require $O(k^2)$ quantum gates if no ancilla are available [BBC$^+$95].

Theorem 2.16 (Implementation of $\wedge_3(X)$ using one ancilla [Mas16])**.**
A $\wedge_3(X)$ gate may be implemented using 1 ancilla qubit that begins and ends in the same arbitrary state and requires a circuit count and depth of 20 Clifford gates and 16 T gates.

Theorem 2.17 (Implementation of $\wedge_k(X)$ using $\lceil \frac{k-2}{2} \rceil$ ancilla [Mas16])**.**
Let $k \geq 4$. A $\wedge_k(X)$ gate may be implemented using $\lceil \frac{k-2}{2} \rceil$ ancilla qubits, which begin and end in the same arbitrary state. This circuit requires a circuit count and depth of $12k - 18$ Clifford gates and a circuit count and depth of $8k - 8$ T gates.

Theorem 2.18 (Implementation of $\wedge_k(X)$ with one ancilla (Lemma 7.3 [BBC$^+$95]))**.**
Let $k \geq 3$ and $r \in \{2, \ldots, k-1\}$. A $\wedge_k(X)$ gate can be implemented with one ancilla qubit which begins and ends in the same arbitrary state and the serial application of two $\wedge_r(X)$ gates and two $\wedge_{k-r+1}(X)$ gates, assuming there are sufficient ancilla to implement these smaller gates.

Proof. Let $\wedge_k(X)$ be defined as acting upon the basis state $|x_1 \ldots x_r x_{r+1} \ldots x_k\rangle |b\rangle |y\rangle$, where $|b\rangle$ is the ancilla bit in an arbitrary state, $|y\rangle$ is the target qubit and $b, y \in \{0, 1\}$. If we apply one $\wedge_r(X)$ gate with the controls set to be $|x_1 \ldots x_r\rangle$ and the target set to be the ancilla bit $|b\rangle$, then the computational basis state is now

$$|x_1 \ldots x_r x_{r+1} \ldots x_k\rangle |b \oplus (x_1 \wedge \cdots \wedge x_r)\rangle |y\rangle . \tag{2.54}$$

Applying a single $\wedge_{k-r+1}(X)$ gate with the controls set to be $|x_{r+1} \ldots x_k\rangle$ and the ancilla qubit, with the target qubit set to be the output qubit of the $\wedge_k(X)$ gate then leaves us in the state

$$|x_1 \ldots x_r x_{r+1} \ldots x_k\rangle |b \oplus (x_1 \wedge \cdots \wedge x_r)\rangle |y \oplus (b \wedge x_{r+1} \wedge \cdots \wedge x_k) \oplus (x_1 \wedge \cdots \wedge x_r \wedge x_{r+1} \wedge \cdots \wedge x_k)\rangle , \tag{2.55}$$

owing to the distributive property of $\oplus$ and $\wedge$. Applying another $\wedge_r(X)$ gate with the controls set to be $|x_1 \ldots x_r\rangle$ and the target set to be the ancilla qubit then resets the ancilla, giving us

$$|x_1 \ldots x_r x_{r+1} \ldots x_k\rangle |b\rangle |y \oplus (b \wedge x_{r+1} \wedge \cdots \wedge x_k) \oplus (x_1 \wedge \cdots \wedge x_r \wedge x_{r+1} \wedge \cdots \wedge x_k)\rangle . \tag{2.56}$$

The final $\wedge_{k-r+1}(X)$ gate is then applied, with the controls set to be $|x_{r+1} \ldots x_k\rangle$ and the ancilla bit, with the target being the output bit. This leaves us in the state

$$|x_1 \ldots x_r x_{r+1} \ldots x_k\rangle |b\rangle |y \oplus (x_1 \wedge \cdots \wedge x_r \wedge x_{r+1} \wedge \cdots \wedge x_k)\rangle , \tag{2.57}$$

completing the implementation of the $\wedge_k(X)$ gate as described in Theorem 2.18. □

Theorem 2.19 follows from one suggested by Dr. Dmitri Maslov in personal communication.

Theorem 2.19 (Clifford+T cost of implementing $\wedge_k(X)$ using one ancilla qubit)**.**
Let $k \geq 7$. A $\wedge_k(X)$ gate can be implemented using one ancilla qubit that starts and ends in the same arbitrary state and the serial application of $24k - 48$ Clifford gates and $16k - 16$ T gates.

Proof. As in [BBC+95], we set $m_1 = \lceil \frac{k+2}{2} \rceil$ and $m_2 = k - m_1 + 1$. We can then use two $\wedge_{m_1}(X)$ gates and two $\wedge_{m_2}(X)$ gates and one ancilla qubit as in Theorem 2.18 to implement the $\wedge_k(X)$ gate. As these gates are operated in serial, the $\wedge_{m_1}(X)$ gate has $k - m_1 + 1$ qubits available at all times that can act as ancilla qubits — as $k - m_1 + 1 - \lceil \frac{m_1 - 2}{2} \rceil \geq 0$ for $k \geq 7$ (see below), we will always have enough ancilla qubits to enable the action of $\wedge_{m_1}(X)$. The $\wedge_{m_2}(X)$ gate has $k - m_2 + 1$ qubits available at all times and as $k - m_2 + 1 - \lceil \frac{m_2 - 2}{2} \rceil \geq 0$ for $k \geq 7$ (see below), we always have enough ancilla qubits to enable the action of $\wedge_{m_1}(X)$. The result follows.

That $l_1(k) = k - \lceil \frac{k+2}{2} \rceil + 1 - \left\lceil \frac{\lceil \frac{k+2}{2} \rceil - 2}{2} \right\rceil \geq 0$ for $k \geq 7$ can be seen by using the identity $\left\lceil \frac{\lceil \frac{x}{y} \rceil}{n} \right\rceil = \left\lceil \frac{x}{yn} \right\rceil$, which holds for $x, y \in \mathbb{R}$ and $n \in \mathbb{N}$ [GKPL89]. Using this identity we have that

$$l_1(k) \geq \frac{k}{2} - 1 - \left\lceil \frac{k-2}{4} \right\rceil \geq \frac{k}{2} - 2 - \frac{k-2}{4} = \frac{k+2}{4} - 2, \tag{2.58}$$

which holds when $k \geq 7$.

That $l_2(k) = k - (k - \lceil \frac{k+2}{2} \rceil + 1) + 1 - \left\lceil \frac{k - \lceil \frac{k+2}{2} \rceil + 1 - 2}{2} \right\rceil = \lceil \frac{k+2}{2} \rceil - \left\lceil \frac{k - \lceil \frac{k+2}{2} \rceil - 1}{2} \right\rceil \geq 0$ for $k \geq 7$ comes from the fact that

$$l_2(k) \geq \frac{k+2}{2} - \frac{k - \frac{k+2}{2} - 1}{2} - 1 = \frac{k}{4} + 1, \tag{2.59}$$

which is clearly satisfied when $k \geq 7$. □

2.3.2.1 The SWAP gate

The SWAP gate acts upon the two qubit computational basis state $|x_1\rangle |x_2\rangle$, where $x_i \in \{0, 1\}$ and can be interpreted as simply swapping the values of these two bits, so that

$$SWAP |x_1\rangle |x_2\rangle \mapsto |x_2\rangle |x_1\rangle . \tag{2.60}$$

The SWAP gate has the following quantum circuit representation and can be implemented by three $CNOT$ gates. The SWAP gate has many uses in quantum computation — in particular it

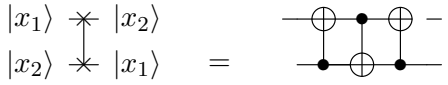


Figure 2-8: The SWAP gate.

can be used to bypass problems with the topology of connections between qubits or it can simply be used to relabel qubits (though this may as well be done by simply relabelling the wires).

2.3.2.2 The $\wedge_1(SWAP)$ (Fredkin) gate

The $\wedge_1(SWAP)$ (also known as the *Fredkin*) gate acts upon the three qubit basis state $|x_1\rangle |x_2x_3\rangle$, where $x_i \in \{0,1\}$, and can be interpreted as performing a controlled-SWAP operation on the the second two qubits if and only if $x_1 = 1$, so that

$$\wedge_1(SWAP) |x_1\rangle |x_2x_3\rangle \mapsto \begin{cases} |x_1\rangle |x_3x_2\rangle & \text{if } x_1 = 1 \\ |x_1\rangle |x_2x_3\rangle & \text{if } x_1 = 0. \end{cases} \tag{2.61}$$

We will later use the $\wedge_1(SWAP)$ gate in the implementation of a quantum counter, which was first suggested by Schwabe and Westerbaan [SW16] and has applications to low-qubit implementations of quantum oracles, which we discuss in Section 2.4.3.

2.3.2.3 A conversion dictionary between reversible circuits and quantum circuits

We therefore have a set of quantum gates $\{X, \wedge_1(X), \wedge_2(X), \wedge_k(X), SWAP, \wedge_1(SWAP)\}_{k=3}^{<\infty}$ which act purely upon the computational basis states and can be used to implement quantum evaluations. The set $\{X, \wedge_1(X), \wedge_2(X)\}$ is clearly equivalent to the universal boolean logic gate set $\{\neg, \oplus, \wedge\}$ and the other quantum gates we describe $\wedge_k(X), SWAP$ and $\wedge_1(SWAP)$ are simply tools in that they provide efficient quantum realisations of useful circuits.

By the above, it is clear that designing a basic quantum evaluation based upon a classical circuit-design requires no more theory than that required for designing reversible classical circuits using these components. Given a suitable framework or formula, such as Equation (3.36) which describes the circuit-complexity of Grover's algorithm, it is then a simple matter to compute the cost of executing Grover's algorithm if we have a dictionary to assign each reversible gate a cost in terms of quantum-gates relative to a universal gate set (see Section 2.2.1) of choice. In this thesis our fixed choice of univeral quantum gate set is the Clifford+T gate set (see Section 2.2.2) and we provide such a dictionary in Table 2.1 below. In this thesis we follow this methodology, designing reversible circuits and translating the costs via Table 2.1.

	Clifford gates	T gates	Total size	T depth	Total depth	Qubits/ ancillae	Notes
X	1	0	1	0	1	1/0	We include $X \in$ Clifford
Z	1	0	1	0	1	1/0	We include $Z \in$ Clifford
H	1	0	1	0	1	1/0	
$\wedge_1(X)$	1	0	1	0	1	2/0	$CNOT$ gate
$\wedge_2(X)$	10	7	17	3	10	3/0	Toffoli gate [Sel13]
$\wedge_3(X)$	20	16	36	20	36	4/1	(ancilla in state $\|x\rangle$) Theorem 2.16
$\wedge_4(X)$	30	24	54	24	54	5/1	(ancilla in state $\|x\rangle$) Theorem 2.17
$\wedge_5(X)$	42	32	74	32	74	6/2	(ancilla in state $\|xx\rangle$) Theorem 2.17
$\wedge_6(X)$	54	40	94	40	94	7/2	(ancilla in state $\|xx\rangle$) Theorem 2.17
$\wedge_k(X)$	$24k-48$	$16k-16$	$40k-64$	$16k-16$	$40k-64$	$k+1/1$	$k \geq 7$ (ancilla in state $\|x \dots x\rangle$) Theorem 2.18
$SWAP$	3	0	3	0	3	2/0	—
$\wedge_1(\text{SWAP})$	10	7	17	5	10	3/0	Fredkin gate [AMMR13]

Table 2.1: Useful circuit costs in the Clifford+T universal quantum gate set.

We therefore have all the tools we need to implement and derive the cost of quantum evaluations and have discussed the implementation of the quantum phase oracle $\mathcal{O}_\chi$ by these methods.

Ultimately it is important to understand that it is the *action* of the quantum phase oracle that it important — if we can implement it via more efficient quantum algorithms or other methods, then all the better. As we will later see, the execution cost or circuit-complexity of the implementation of the quantum phase oracle has a very real impact upon Grover's algorithm (see Section 3.2.1). It is this overhead in Grover's algorithm that stems from implementing quantum phase oracles via quantum evaluations and the analysis of methods that can be used to overcome these costs that motivates this thesis.

2.3.3 The implementation of $-\mathcal{O}_{0_n}(\phi)$

We briefly discuss the implementation of one specific unitary operator that will be required for the theory of amplitude amplification (see Theorem 3.6), that of $-\mathcal{O}_{0_n}(\phi)$, which is simply a quantum phase oracle with a negative phase. The negative phase is not technically required as it simply adds an invisible global phase, but is relatively cheap to implement.

Definition 2.20 (The boolean-function $0_n : \{0,1\}^n \longrightarrow \{0,1\}$)**.**
We define $0_n : \{0,1\}^n \longrightarrow \{0,1\}$, the indicator function for the zero-bitstring, by the function

$$0_n(x) \mapsto \begin{cases} 1 & \text{if } x = 0^n \\ 0 & \text{if } x \neq 0^n. \end{cases} \tag{2.62}$$

The boolean function $0_n : \{0,1\}^n \longrightarrow \{0,1\}$ in conjunction with a phase $0 \leq \phi < 2\pi$ naturally defines the quantum phase oracle $\mathcal{O}_{0_n}(\phi)$.

Theorem 2.21 (Clifford+T Implementation cost of $-\mathcal{O}_{0_n}(\phi)$)**.**
Let $n \geq 8$ and let $0 \leq \phi < 2\pi$ and $0_n : \{0,1\}^n \longrightarrow \{0,1\}$ be as in Definition 2.20. The unitary operator $-\mathcal{O}_{0_n}(\phi)$ can be implemented using a quantum circuit which requires one R_ϕ gate in addition to at most $50n - 142$ Clifford gates and $32n - 64$ T gates for a total of $82n - 206$ Clifford+T gates. This quantum circuit has a T gate depth of $32n - 64$ and a total Clifford+T depth of $80n - 206$, in addition to the depth required by the R_ϕ gate. This approach requires 1 ancilla qubit in any state, for a total of $n + 1$ qubits.

Proof. If we apply $n - 1$ X gates in parallel to the first $n - 1$ qubits and use a $\wedge_{n-1}(X)$ gate with these qubits as the controls and the final qubit as the target, then the n^{th} bit of each computational basis state is $|1\rangle$ if and only if the initial state was $|0^n\rangle$. We need only apply a R_ϕ gate to the final qubit to obtain the action of the conditional phase-kickback before restoring the computational basis state to its original state by applying $\wedge_{n-1}(X)$ and $n-1$ X gates again.

The negative phase can easily be added to this operator via applying the sequence of gates $XZXZ$ to a qubit that is not being operated on for at least four time-steps. It is easily checked that this sequence maps $|x\rangle \mapsto -|x\rangle$. The costs of this algorithm follow from Table 2.1. □

We can implement the unitary operator $-\mathcal{O}_{0_n} = -\mathcal{O}_{0_n}(\pi)$ via an approach similar to the ancilla based oracle approach for a slightly smaller cost.

Theorem 2.22 (Implementation cost of $-\mathcal{O}_{0_n} = -\mathcal{O}_{0_n}(\pi)$)**.**
Let $n \geq 8$ and $0_n : \{0,1\}^n \longrightarrow \{0,1\}$ be as in Definition 2.20. The unitary operator $-\mathcal{O}_{0_n}$ can be implemented using a quantum circuit which requires at most $26n-66$ Clifford gates and $16n-32$ T gates. This quantum circuit has a T gate depth of $16n-32$ and a total depth of $40n-102$ Clifford+T gates. This approach requires 1 ancilla qubit in any state, for a total of $n+1$ qubits.

Proof. If we apply an X gate followed by a Hadamard gate to the n^{th} qubit, then we have that this qubit will be in the state $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$ if it was initially $|0\rangle$ and in the state $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ if it was initially $|1\rangle$. If an X gate is applied to each of the first $n-1$ qubits and a $\wedge_{n-1}(X)$ gate uses these qubits as controls with the n^{th} qubit as a target, then it is plain that as the state $|+\rangle$ is invariant with regards to the X gate and that the phase will only be altered if the initial state was $|0^n\rangle$. The H gate and n X gates can then be applied once more to restore each computational basis state to their original values. Using this construction in conjunction with the implementation of $\wedge_{n-1}(X)$ as given in Theorem 2.19 gives the stated costs.

As in the proof of Theorem 2.21, the negative phase can easily be added to this operator via applying the sequence of gates $XZXZ$ to a qubit that is not being operated on for at least four time-steps. The costs of this algorithm follow from Table 2.1. □

It will be useful to introduce some notation to describe the cost of quantum oracles and algorithms. We describe the cost of quantum algorithms relative to the execution of subroutines that must be executed in serial. In this way we can derive a cost equation for a quantum algorithm and later substitute the cost-metric we are interested in, which may be the circuit-size or circuit-depth relative to either the total number of quantum gates or a specific quantum gate.

Definition 2.23 (Cost notation)**.**
We use the following *execution cost notation for amplitude amplification* to represent the cost (where cost can be circuit-size or circuit-depth) of a quantum circuit

- $E_{\mathcal{A}}$ represents the cost of executing an arbitrary quantum algorithm or gate $\mathcal{A}$.
- $E_{\bar{n}}$ represents the cost of executing $-\mathcal{O}_{0_n}$.
- $E_{\mathcal{O}_\chi}$ represents the cost of executing the quantum phase oracle $\mathcal{O}_\chi$.
- $E_{\mathcal{O}_\chi^{(b)}}$ represents the cost of executing the quantum bit oracle $\mathcal{O}_\chi^{(b)}$.
- $E_{\mathcal{E}_h}$ represents the cost of executing the quantum evaluation $\mathcal{E}_h$ or $\mathcal{E}_h^\dagger$.

2.4 Quantum oracle designs for the preimage search problem

In this section we discuss several basic quantum bit oracle design patterns that can be applied to the single-target preimage search problem (see Definition 1.2) defined by $h : \{0,1\}^n \longrightarrow \{0,1\}^m$, $Y_h = \{y_h\} \subseteq \{0,1\}^m$ and $M_h = h^{-1}(y_h)$. The first design relies upon no structure in the function $h : \{0,1\}^n \longrightarrow \{0,1\}^m$ and is very simple, whilst the second design focuses upon exploiting structure in the function $h : \{0,1\}^n \longrightarrow \{0,1\}$ in order to lower the number of qubits required at the expense of the circuit-complexity of the quantum bit oracle.

2.4.1 A direct approach

Given a quantum evaluation $\mathcal{E}_h$ which requires w ancilla bits for working memory, a quantum bit oracle for the single-target preimage search problem can be implemented by the serial application of $\mathcal{E}_h$, at most m X gates executed in parallel, one $\wedge_m(X)$ gate, at most m X gates executed in parallel and one application of $\mathcal{E}_h^\dagger$. This approach requires $n + w + m + 1$ qubits and has an execution cost of at most $2E_{\mathcal{E}_h} + 2E_{X^{\otimes m}} + \wedge_m(X)$. This is easily seen as executing $\mathcal{E}_h$ computes

$$|x\rangle |0^w\rangle |0^m\rangle |0\rangle \mapsto |x\rangle |g(x)\rangle |h(x)\rangle |0\rangle \tag{2.63}$$

and we can then use at most m X gates to compute

$$|x\rangle |g(x)\rangle |h(x) \oplus y_h \oplus 1^m\rangle |0\rangle \tag{2.64}$$

before applying one $\wedge_m(X)$ gate to compute the output bit, so that we obtain

$$|x\rangle |g(x)\rangle |h(x) \oplus y_h \oplus 1^m\rangle \ |h(x) \stackrel{?}{=} y_h\rangle \,. \tag{2.65}$$

Applying at most m X gates and executing $\mathcal{E}_h^\dagger$ again then clearly returns us to the state

$$|x\rangle |0^w\rangle |0^m\rangle \ |h(x) \stackrel{?}{=} y_h\rangle \,, \tag{2.66}$$

which gives us the action of the quantum bit oracle for the single-target preimage search problem.

2.4.2 A low-memory approach

An alternative approach is possible if the problem admits a *constraint-based decomposition*. This approach is a generalisation of the method introduced by Schwabe and Westerbaan [SW16] for an approach to designing a quantum bit oracle for the $\mathcal{MQ}$ problem over $\mathbb{F}_2$ which requires $n + \lceil \log_2(m+1) \rceil + 2$ qubits for a quantum oracle using this design pattern compared to the $n+m+2$ qubits that the depth optimal approach described in Section 2.4.1. We generalise their approach in this section, then give concrete oracle costs in Sections 2.5 and 2.6.

Definition 2.24 (Constraint based decomposition of χ)**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $k \geq 2$. If we know the existence of $\chi_1, \ldots, \chi_k : \{0,1\}^n \longrightarrow \{0,1\}$ such that each $\chi_i \neq \chi_j$ for $i \neq j$, no χ_i is constant and

$$\chi(x) = \chi_1(x) \wedge \cdots \wedge \chi_k(x), \tag{2.67}$$

then we say that χ admits a *constraint based decomposition*.

The constraint based decomposition is simply a special-case of those traditionally exploited in Constraint Satisfaction Problems (CSP), as each χ_i may involve all variables whereas CSP constraints often involve only a subset of variables. Each χ_i has both an associated implementation cost for the corresponding quantum bit oracle $\mathcal{O}_{\chi_i}^{(b)}$ and evaluation $\mathcal{E}_{\chi_i}$, which are both assumed to require w_i ancilla bits to compute and a set of $M_{\chi_i} = |\chi_i^{-1}(1)|$ preimages.

It may hold that $E_{\mathcal{O}_{\chi_i}^{(b)}} \approx E_{\mathcal{O}_{\chi_j}^{(b)}}$ (also $E_{\mathcal{E}_{\chi_i}} \approx E_{\mathcal{E}_{\chi_j}}$) and $M_{\chi_i} \approx M_{\chi_j}$ for $i \neq j$ or that these values will be very different. In the cases we examine, that of the $\mathcal{MQ}$ problem over $\mathbb{F}_2$ and cryptanalysis of the Advanced Encryption Standard, it will hold that the execution costs will be nearly identical and whilst $M_{\chi_i} \neq M_{\chi_j}$, they will follow the same probability distribution.

Given an instance of a search problem exhibiting a constraint-based decomposition, we could simply treat the decomposition as an instance of the single-target preimage search problem where $h : \{0,1\}^n \longrightarrow \{0,1\}^k$ is defined by $h(x) \mapsto \chi(x)$ and $Y_h = \{1^k\}$. If we use the quantum evaluations $\mathcal{E}_{\chi_1}, \ldots, \mathcal{E}_{\chi_k}$, each of which require w_i ancilla qubits to compute, then such an approach requires $n + \sum_{i=1}^{k} w_i + k + 1$ qubits and has a cost of $2\sum_{i=1}^{k} E_{\mathcal{E}_{\chi_i}} + E_{\wedge_k(X)}$ as we compute

$$|x\rangle |0^{w_1}\rangle \ldots |0^{w_k}\rangle \ |0^k\rangle |0\rangle \mapsto |x\rangle |g_1(x)\rangle \ldots |g_k(x)\rangle |\chi_1(x) \ldots \chi_k(x)\rangle |0\rangle \tag{2.68}$$

and then use a single $\wedge_k(X)$ gate to compute the output bit

$$|x\rangle |g_1(x)\rangle \ldots |g_k(x)\rangle |\chi_1(x) \ldots \chi_k(x)\rangle |\chi_1(x) \wedge \cdots \wedge \chi_k(x)\rangle \tag{2.69}$$

before applying $\mathcal{E}_{\chi_1}^\dagger, \ldots, \mathcal{E}_{\chi_k}^\dagger$ to leave us in the state

$$|x\rangle |0^{w_1}\rangle \ldots |0^{w_k}\rangle \ |0^k\rangle |\chi_1(x) \wedge \cdots \wedge \chi_k(x)\rangle . \tag{2.70}$$

We next examine how we can reduce the number of ancilla qubits at the cost of the circuit-complexity via using quantum bit oracles instead of quantum evaluations.

Setting $\bar{w} = \max\{w_i\}_{i=1}^k$, we can execute $\mathcal{O}_{\chi_1}^{(b)}, \ldots, \mathcal{O}_{\chi_{k-1}}^{(b)}, \mathcal{E}_{\chi_k}$ in serial, giving us the state

$$|x\rangle \ |0^{\bar{w}}\rangle \ |0^k\rangle |0\rangle \mapsto |x\rangle \ |g_k(x)\rangle |\chi_1(x) \ldots \chi_k(x)\rangle |0\rangle \tag{2.71}$$

as the $\bar{w}$ ancilla qubits are enough to compute any one of these unitary operators. A single $\wedge_k(X)$ gate is then enough to compute the output bit of the quantum oracle $\mathcal{O}_{\chi}^{(b)}$, giving us

$$|x\rangle \ |g_k(x)\rangle |\chi_1(x) \ldots \chi_k(x)\rangle |\chi_1(x) \wedge \cdots \wedge \chi_k(x)\rangle . \tag{2.72}$$

Finally, we need only execute $\mathcal{E}_{\chi_k}^\dagger, \mathcal{O}_{\chi_{k-1}}^{(b)}, \ldots, \mathcal{O}_{\chi_1}^{(b)}$ in serial to obtain the state

$$|x\rangle \ |0^{\bar{w}}\rangle \ |0^k\rangle |\chi_1(x) \wedge \cdots \wedge \chi_k(x)\rangle , \tag{2.73}$$

giving us the action of the quantum oracle $\mathcal{O}_{\chi}^{(b)}$. This approach requires $n + \max\{w_i\}_{i=1}^k + k + 1$ qubits to implement, but has a total cost of

$$2\sum_{i=1}^{k-1} E_{\mathcal{O}_{\chi_i}^{(b)}} + 2E_{\mathcal{E}_{\chi_k}} + E_{\wedge_k(X)}. \tag{2.74}$$

This method is inherently serial and, as $E_{\mathcal{O}_{\chi_i}^{(b)}} \approx 2E_{\mathcal{E}_{\chi_i}}$ for the problems we examine, this approach approximately doubles the quantum circuit-size of the quantum bit oracle over the direct method.

2.4.3 A counter-based approach

Schwabe and Westerbaan's counter-based oracle uses this approach to reduce the number of ancilla bits used for memory and additionally reduces the dependence on requiring k qubits to store $|\chi_1(x)\dots\chi_k(x)\rangle$, requiring only $n + \max\{w_i\}_{i=1}^k + \lceil \log_2(k+1)\rceil + 1$ qubits. The counter-based approach consists of using a *counter register* consisting of c qubits, which records only how many of $\chi_1(x),\dots,\chi_k(x)$ have been satisfied — not which ones. Assuming that $|z_1\dots z_c\rangle$ represents the counter register, we have that the procedure consists of the following:

- For $i = 1,\dots,k$
  - Execute $\mathcal{E}_{\chi_i}$ to compute $|x\rangle|g_i(x)\rangle|\chi_i(x)\rangle|z_1\dots z_c\rangle$.
  - Perform an increment of the counter controlled upon $|\chi_i(x)\rangle$.
  - Execute $\mathcal{E}_{\chi_i}^\dagger$ to restore the state to $|x\rangle|0^{\bar{w}}\rangle|0\rangle|z'_1\dots z'_c\rangle$.
- Via applying at most c X gates, a single $\wedge_c(X)$ gate and at most another c X gates, we output $|1\rangle$ if and only if the counter has been incremented k times.
- For $i = k,\dots,1$
  - Execute $\mathcal{E}_{\chi_i}$ to compute $|x\rangle|g_i(x)\rangle|\chi_i(x)\rangle|z_1\dots z_c\rangle$.
  - Perform a decrement of the counter controlled upon $|\chi_i(x)\rangle$.
  - Execute $\mathcal{E}_{\chi_i}^\dagger$ to restore the state to $|x\rangle|0^{\bar{w}}\rangle|0\rangle|z'_1\dots z'_c\rangle$.

The counter-register itself can be implemented using only $c = \lceil \log_2(k+1)\rceil$ qubits to represent a maximum of k possible increments. The counter-register encodes a number from 0 to k by allowing the c bits of each computational basis state to represent a non-zero element of the ring $\mathbb{F}_2[Z]/(p(Z))$, where $p(Z)$ is a primitive polynomial of degree c. In this way we have that

$$|z_1\dots z_c\rangle \leftrightarrow z_1Z^0 + z_2Z^1 + \cdots + z_cZ^{c-1} \tag{2.75}$$

By standard properties of this ring, the action of multiplication by Z on a non-zero element of $\mathbb{F}_2[Z]/(p(Z))$ induces a permutation of the $2^c - 1$ non-zero elements of this ring. Hence if we start the counter register in the state $|z_1\dots z_c\rangle \neq |0^c\rangle$ and perform a controlled multiplication by Z for increments or Z^{-1} for decrements, then the counter can be easily implemented. The exact element that corresponds to the counter register being incremented k times is easily precomputed by classically computing the value of the counter-register which corresponds to the $\mathbb{F}_2[Z]/(p(Z))$ interpretation $Z^k \cdot \left(z_1Z^0 + z_2Z^1 + \cdots + z_cZ^{c-1}\right)$.

We first discuss how an increment of the counter can be performed, noting that if we wish to decrement, then we simply need run this procedure in reverse. An increment is simply multiplication of a ring element by Z and is done in two stages. First, a multiplication by Z is performed in the ring $\mathbb{F}_2[Z]/(Z^c+1)$, which corresponds to mapping

$$|z_1z_2\dots z_{c-1}z_c\rangle \mapsto |z_cz_1\dots z_{c-2}z_{c-1}\rangle\,. \tag{2.76}$$

This can be performed via the serial application of $c-1$ SWAP gates. An addition of the element $p(Z)-(Z^c+1)$ is then performed, controlled on the value of z_c. This can be performed via at most $c-1$ $\wedge_1(X)$ gates. As a controlled SWAP gate is a $\wedge_1(SWAP)$ or Fredkin gate and a controlled $\wedge_1(X)$ gate is simply a $\wedge_2(X)$ or Toffoli gate, a single controlled increment or decrement therefore costs $(c-1)\cdot(E_{\wedge_1(SWAP)}+E_{\wedge_2(X)})$.

As we need not uncompute or compute the working memory $|g_k(x)\rangle$ for the correctness to be preserved and we can leave the most expensive quantum evaluation til last, we can save one execution of $\mathcal{E}_{\chi_k}^\dagger$ and $\mathcal{E}_{\chi_k}$. A counter-based quantum-bit oracle therefore requires $n+\max\{w_i\}_{i=1}^k+\lceil\log_2(k+1)\rceil+1$ qubits and has an execution cost of at most

$$4\sum_{i=1}^{k-1}E_{\mathcal{E}_{\chi_i}}+2E_{\mathcal{E}_{\chi_k}}+2k(\lceil\log_2(k+1)\rceil-1)\cdot\left(E_{\wedge_1(SWAP)}+E_{\wedge_2(X)}\right)+2E_{X^{\otimes\lceil\log_2(k+1)\rceil}}+E_{\wedge_{\lceil\log_2(k+1)\rceil}(X)}. \quad (2.77)$$

The counter-based approach therefore requires approximately double the circuit-size of the direct approach, but is advantageous in terms of the number of qubits required as it both allows us to reuse the ancilla qubits used to compute and record the fulfillment of the constraints $\chi_1,\ldots,\chi_k$.

2.5 Quantum evaluations and bit oracles for the $\mathcal{MQ}$ problem

We now examine how the above frameworks impact upon the design of a quantum bit oracle for the $\mathcal{MQ}$ problem over $\mathbb{F}_2$. We first discuss the design principles used by Schwabe and Westerbaan, relating how they fit into the above framework.

2.5.1 Previous work by Schwabe and Westerbaan

Schwabe and Westerbaan describe an instance of the $\mathcal{MQ}(\mathbb{F}_2,n,m)$ in terms of a *cube* of coefficients $\left(\lambda_{i,j}^{(k)}\right)$, where $1\leq i,j\leq n$ and $1\leq k\leq m$ and a *vector* $v=(v_1,\ldots,v_m)\in\mathbb{F}_2^m$. These describe a system of degree-two equations over $\mathbb{F}_2$ whose solution is the vector $(x_1,\ldots,x_n)\in\mathbb{F}_2^n$ such that for $k=1,\ldots,m$ it holds that

$$\sum_{1\leq i,j\leq n}\lambda_{i,j}^{(k)}x_ix_j=0. \quad (2.78)$$

A transformation is then applied in a classical preprocessing phase to convert this cube into form they refer to as a *convenient transformation*. Their core results are in the design of the quantum bit oracle and we present these results using a standard and more intuitive representation of a quadratic equation over $\mathbb{F}_2$. It is clear that any degree-two equation $f^{(i)}\in\mathbb{F}_2[x_1,\ldots,x_n]$ has the representation

$$f^{(k)}(x_1,\ldots,x_n)=\sum_{1\leq i<j\leq n}a_{i,j}^{(k)}x_ix_j+\sum_{1\leq i\leq n}b_i^{(k)}x_i+c^{(k)}, \quad (2.79)$$

owing to the fact that $x_jx_i=x_ix_j$ and $x_ix_i=x_i$. We will present their method in relation to this representation, which does not change the essential nature of their results.

Their convenient transformation essentially reduces the cube representation to that of (2.79), ensuring that the commutative property $x_i x_j = x_j x_i$ and the identity $x_i^2 = x_i$ is taken into account and has the consequence that instead of evaluating and checking whether

$$f^{(1)}(x_1, \ldots, x_n) = \cdots = f^{(m)}(x_1, \ldots, x_n) = 0, \tag{2.80}$$

the equivalent condition of evaluating and checking whether

$$\tilde{f}^{(1)}(x_1, \ldots, x_n) = \cdots = \tilde{f}^{(m)}(x_1, \ldots, x_n) = 1 \tag{2.81}$$

is satisfied is performed, where

$$\tilde{f}^{(k)}(x_1, \ldots, x_n) = f^{(k)}(x_1, \ldots, x_n) + c^{(k)} + 1. \tag{2.82}$$

It is easily seen that the computational resources required to evaluate $f^{(k)}(x_1, \ldots, x_n)$ and $\tilde{f}^{(k)}(x_1, \ldots, x_n)$ are essentially identical. The quantum bit oracle designs are then those given in Sections 2.4.2 and 2.4.3, with $k = m$ and each constraint defined by $\chi_i(x_1 \ldots x_n) = \tilde{f}^{(k)}(x_1, \ldots, x_n)$. We have simply generalised their methods in these sections. All that remains is to examine their method of implementing the quantum bit oracles $\mathcal{O}_{f^{(i)}}$ and quantum evaluations $\mathcal{E}_{f^{(i)}}$. We will then easily derive the total cost of their quantum bit oracles $\mathcal{O}_{\chi}^{(b)}$ for the $\mathcal{MQ}$ problem over $\mathbb{F}_2$.

A naive approach for creating these primitives would be evaluating $f^{(k)}(x_1, \ldots, x_n)$ via the addition of each term of $f^{(k)}(x_1, \ldots, x_n)$ to an equation register (consisting of a single qubit) which holds the result of the evaluation. This would require at most 1 X gate for the addition of $c^{(k)}$, at most n $\wedge_1(X)$ gates for the addition of the $b_i^{(k)} x_i$ terms and at most $\frac{n^2-n}{2}$ $\wedge_2(X)$ gates. Whilst this can be done using no ancilla bits, this proves to be a large number of $\wedge_2(X)$ gates, which as we recall (see Section 2.3.1.3) require 10 Clifford gates and 7 T gates to implement. Schwabe and Westerbaan instead suggest an alternative approach which requires only n $\wedge_2(X)$ gates per equation, with the majority of the work being done via $\wedge_1(X)$ gates. The basic strategy that Schwabe and Westerbaan employ for evaluating a single equation $f^{(k)}(x_1, \ldots, x_n)$ is to observe the decomposition

$$f^{(k)}(x_1, \ldots, x_n) = c^{(k)} + \sum_{i=1}^{n} x_i y_i^{(k)} \qquad \text{where} \qquad y_i^{(k)} = b_i^{(k)} + \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_j, \tag{2.83}$$

using a single ancilla qubit to act as a temporary storage register to compute each $|y_i^{(k)}\rangle$, which starts and ends in the state $|0\rangle$. The procedure consists of the following.
Add $c^{(k)}$ to the output register. Then for $i = 1, \ldots, n$:

- Compute $|y_i^{(k)}\rangle$ using the ancilla qubit using at most 1 X gate and $n - i$ $\wedge_1(X)$ gates.
- Add $x_i y_i^{(k)}$ to the register which will store $|f^{(k)}(x_1, \ldots, x_n)\rangle$ via a single $\wedge_2(X)$ gate.
- Uncompute $|y_i^{(k)}\rangle$ using the ancilla qubit using at most 1 X gate and $n - i$ $\wedge_1(X)$ gates.

The above procedure clearly implements the quantum bit oracle $\mathcal{O}_{f^{(k)}}^{(b)}$ and requires $n+2$ qubits.

In terms of the total number of gates, the procedure therefore requires at most n $\wedge_2(X)$ gates, $n^2 - n$ $\wedge_1(X)$ gates and $2n+1$ X gates. There construction therefore has the following requirements

	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits
$\mathcal{O}^{(b)}_{\tilde{f}^{(k)}}$	$n^2+11n+1$	$7n$	$n^2+18n+1$	$3n$	$n^2+10n+1$	$n+2$

Table 2.2: Clifford+T gate costs for Schwabe and Westerbaan's quantum bit oracle $\mathcal{O}^{(b)}_{\tilde{f}^{(k)}}$.

This primitive can then be used in conjunction with the constraint-based quantum bit oracle designs given in Sections 2.4.2 and 2.4.3. We note that Schwabe and Westerbaan describe their circuit in terms of only the quantum bit oracles $\mathcal{O}^{(b)}_{\tilde{f}^{(i)}}$, hence their versions of the cost Equations (2.74) and (2.77) are respectively

$$2\sum_{i=1}^{m} E_{\mathcal{O}^{(b)}_{\tilde{f}^{(i)}}} + E_{\wedge_m(X)} \tag{2.84}$$

and at most

$$4\sum_{i=1}^{m} E_{\mathcal{O}^{(b)}_{\tilde{f}^{(i)}}} + 2m(\lceil \log_2(m+1)\rceil - 1)\cdot\left(E_{\wedge_1(SWAP)} + E_{\wedge_2(X)}\right) + 2E_{X^{\otimes\lceil\log_2(m+1)\rceil}} + E_{\wedge_{\lceil\log_2(m+1)\rceil}(X)}. \tag{2.85}$$

The difference will be minor in the case of the $\mathcal{MQ}$ problem and Grover's algorithm, as we assume that $m \geq n$ and that the implementation of the quantum evaluation $\mathcal{E}_{f^{(k)}}$ will be identical to the quantum bit oracle $\mathcal{O}^{(b)}_{\tilde{f}^{(k)}}$, save simply not uncomputing the register $|y^{(k)}_n\rangle$. The difference in cost between a quantum evaluation and quantum bit oracle is therefore negligible in this case, but will later be important in Chapter 6, where k will be relatively small and we are dealing with the key-search problem instead. The two constructions therefore have the cost

Method used	Section 2.4.2	Section 2.4.3
#Clifford gates	$2mn^2+22mn+26m-48$	$4mn^2+44mn+(40\lceil\log_2(m+1)\rceil-36)m+26\lceil\log_2(m+1)\rceil-48$
#T gates	$14mn+16m-16$	$28mn+(28\lceil\log_2(m+1)\rceil-14)m+16\lceil\log_2(m+1)\rceil-16$
#Total gates	$2mn^2+36mn+42m-64$	$4mn^2+72mn+(68\lceil\log_2(m+1)\rceil-64)m+42\lceil\log_2(m+1)\rceil-64$
T depth	$6mn+16m-16$	$12mn+16(\lceil\log_2(m+1)\rceil-1)m+16\lceil\log_2(m+1)\rceil-16$
Total depth	$2mn^2+22mn+40m-64$	$4mn^2+44mn+(40\lceil\log_2(m+1)\rceil-40)m+40\lceil\log_2(m+1)\rceil-62$
#Qubits	$n+m+2$	$n+\lceil\log_2(m+1)\rceil+2$

Table 2.3: Clifford+T cost of quantum bit oracles from [SW16] for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$.

We will return to these costs and the full cost of quantum search using Grover's algorithm in conjunction with Schwabe and Westerbaan's quantum bit oracles once we have discussed Grover's algorithm in Section 3.2.1. For now we now turn to the slightly optimised versions of quantum evaluations and quantum bit oracles that we will use throughout this thesis.

2.5.2 Our standard quantum evaluation $\mathcal{E}_{\tilde{f}^{(i)}}$ and quantum bit oracle $\mathcal{O}_{\tilde{f}^{(i)}}$

We make several changes to Schwabe and Westerbaan's representation, which do not impact upon the cost and will later be of benefit when it comes to quantum oracle design. The first change is simply in the representation of the individual equations. Instead of the representation

$$f^{(k)}(x_1,\ldots,x_n) = c^{(k)} + \sum_{i=1}^{n} x_i y_i^{(k)} \qquad \text{where} \qquad y_i^{(k)} = b_i^{(k)} + \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_j, \tag{2.83}$$

we simply change definition of $y_i^{(k)}$ so that we have the representation

$$f^{(k)}(x_1,\ldots,x_n) = c^{(k)} + \sum_{i=1}^{n} x_i y_i^{(k)} \qquad \text{where} \qquad y_i^{(k)} = b_i^{(k)} + \sum_{j=1}^{i-1} a_{j,i}^{(k)} x_j. \tag{2.86}$$

With this representation, the partial sum $c^{(k)} + \sum_{i=1}^{n'} x_i y_i^{(k)}$ where $n' \leq n$, involves only the variables $x_1,\ldots,x_{n'}$. This will later be exploited in Chapter 4. We additionally make a minor change to save one Toffoli gate, noting that

$$x_1 y_1^{(k)} = b_1^{(k)} x_1 \qquad \text{and} \qquad x_2 y_2^{(k)} = b_2^{(k)} x_2 + a_{1,2} x_1 x_2. \tag{2.87}$$

This means that we can add $x_1 y_1^{(k)} + x_2 y_2^{(k)}$ to the register holding the evaluated equation with two $\wedge_1(X)$ gates and a single $\wedge_2(X)$ gate, without modifying the temporary storage register. In comparison, Schwabe and Westerbaan's method requires that we start and end with the temporary storage register in the state $|0\rangle$, giving us a cost to add $x_{n-1} y_{n-1}^{(k)} + x_n y_n^{(k)}$ to the equation register of 4 X gates 4 $\wedge_1(X)$ gates and 2 $\wedge_2(X)$ gates. This is only a minor optimisation, but worth noting. The quantum evaluation $\mathcal{E}_{\tilde{f}^{(k)}}$ corresponding to $f^{(k)}(x_1,\ldots,x_n)$ for $n \geq 2$ is therefore given by the following procedure.

- Add $c^{(k)} + b_1^{(k)} x_i + b_2^{(k)} x_2 + a_{1,2}^{(k)} x_1 x_2$ to the output register via 1 X, 2 $\wedge_1(X)$ and 1 $\wedge_2(X)$ gates.
- For $i = 3,\ldots,n-1$:
  - Compute $|y_i^{(k)}\rangle$ using the ancilla qubit using at most 1 X gate and $i-1$ $\wedge_1(X)$ gates.
  - Add $x_i y_i^{(k)}$ to the register which will store $|f^{(k)}(x_1,\ldots,x_n)\rangle$ via a single $\wedge_2(X)$ gate.
  - Uncompute $|y_i^{(k)}\rangle$ using the ancilla qubit using at most 1 X gate and $i-1$ $\wedge_1(X)$ gates.
- Compute $|y_n^{(k)}\rangle$ using the ancilla qubit using at most 1 X gate and $n-1$ $\wedge_1(X)$ gates.
- Add $x_n y_n^{(k)}$ to the register which will store $|f^{(k)}(x_1,\ldots,x_n)\rangle$ via a single $\wedge_2(X)$ gate.

This procedure performs the required mapping

$$\mathcal{E}_{\tilde{f}^{(k)}} |x_1 \ldots x_n\rangle |0\rangle |0\rangle \mapsto |x_1 \ldots x_n\rangle \; |y_n^{(k)}\rangle \; |\tilde{f}^{(k)}(x_1,\ldots,x_n)\rangle . \tag{2.88}$$

The quantum evaluation $\mathcal{E}_{\tilde{f}^{(k)}}$ therefore requires $n+2$ qubits and at most $n-1$ $\wedge_2(X)$ gates, $n^2 - 2n + 1$ $\wedge_1(X)$ gates and $2n-4$ X gates.

The quantum bit oracle $\mathcal{O}_{\tilde{f}^{(k)}}$ is simply the same procedure, but with the register holding $|y_n^{(k)}\rangle$ uncomputed at the end, which gives us the mapping

$$\mathcal{O}^{(b)}_{\tilde{f}^{(k)}} |x_1 \dots x_n\rangle |0\rangle |0\rangle \mapsto |x_1 \dots x_n\rangle \ |0\rangle \ |\tilde{f}^{(k)}(x_1, \dots, x_n)\rangle \,. \tag{2.89}$$

The qubit requirement for the quantum bit oracle $\mathcal{O}_{\tilde{f}^{(k)}}$ is therefore still $n+2$ qubits and the cost is at most $n-1$ $\wedge_2(X)$ gates, n^2-n $\wedge_1(X)$ gates and $2n-3$ X gates.

These optimisations are minor and make only a constant difference in the cost, but the main point of this section was to provide discussion around the construction of quantum oracles and provide a standard worst-case formula for computing the cost of the quantum evaluation $\mathcal{E}_{\tilde{f}^{(k)}}$, which fits into the quantum oracle design framework. With these in hand we can easily compute the cost of our optimisations. These primitives have the costs

	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits
$\mathcal{E}_{\tilde{f}^{(k)}}$	$n^2+10n-13$	$7n-7$	$n^2+17n-20$	$3n-3$	$n^2+10n-16$	$n+2$
$\mathcal{O}^{(b)}_{\tilde{f}^{(k)}}$	$n^2+11n-13$	$7n-7$	$n^2+18n-20$	$3n-3$	$n^2+11n-16$	$n+2$

Table 2.4: Clifford+T gate costs for the quantum evaluation and bit oracle $\mathcal{O}^{(b)}_{\tilde{f}^{(k)}}$.

This in turn gives us the worst-case cost of the quantum evaluations and quantum bit oracles we will use for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$. We will use these costs for computations later on.

Type/Method used	$\mathcal{O}_\chi^{(b)}$: Section 2.4.2	$\mathcal{O}_\chi^{(b)}$: Section 2.4.3
#Clifford gates	$2mn^2+22mn-2n-2m-48$	$4(m-1)(n^2+10n-12)+2(n^2+10n-12)+2m(\lceil\log_2(m+1)\rceil-1)20+2\lceil\log_2(m+1)\rceil+24\lceil\log_2(m+1)\rceil-48$
#T gates	$14mn+2m-16$	$4(m-1)(7n-7)+2(7n-7)+2m(\lceil\log_2(m+1)\rceil-1)14+16\lceil\log_2(m+1)\rceil-16$
#Total gates	$2mn^2+36mn-2n+26m-88$	$4(m-1)(n^2+17n-19)+2(n^2+17n-19)+2m(\lceil\log_2(m+1)\rceil-1)34+2\lceil\log_2(m+1)\rceil+40\lceil\log_2(m+1)\rceil-64$
T depth	$6mn+10m-16$	$4(m-1)(3n-3)+2(3n-3)+2m(\lceil\log_2(m+1)\rceil-1)8+16\lceil\log_2(m+1)\rceil-16$
Total depth	$2mn^2+22mn+12m-2n-64$	$4(m-1)(n^2+10n-14)+2(n^2+10n-14)+2m(\lceil\log_2(m+1)\rceil-1)20+2+40\lceil\log_2(m+1)\rceil-64$
#Qubits	$n+m+2$	$n+\lceil\log_2(m+1)\rceil+2$

Table 2.5: Clifford+T cost of quantum bit oracles for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$.

2.5.3 A note on using more qubits for temporary memory

We will be predominantly interested in strategies which use few qubits in this thesis, but it is worth noting that if we allow more qubits then the cost of quantum evaluations compared to quantum bit oracles becomes clearer. If we allow $n-2$ temporary storage registers for the for the computation of $|y_3^{(k)}\rangle \dots |y_n^{(k)}\rangle$, then we need not uncompute $|y_i^{(k)}\rangle$ before computing $|y_{i+1}^{(k)}\rangle$. This leads to the costs for the quantum evaluation and quantum bit oracle as given in Table 2.6.

	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits
$\mathcal{E}_{\tilde{f}^{(k)}}$	$\frac{n^2+21n-20}{2}$	$7n-7$	$\frac{n^2+35n-34}{2}$	$3n-3$	$11n-10$	$2n-1$
$\mathcal{O}^{(b)}_{\tilde{f}^{(k)}}$	$n^2+11n+10$	$7n-7$	$n^2+18n-20$	$3n-3$	$12n-10$	$2n-1$

Table 2.6: Clifford+T gate costs for the high-qubit quantum evaluation $\mathcal{E}_{\tilde{f}^{(k)}}$ and bit oracle $\mathcal{O}^{(b)}_{\tilde{f}^{(k)}}$.

2.6 Quantum evaluations and bit oracles for AES-$\{128, 192, 256\}$

The quantum evaluation and quantum bit oracles for the key-search problem for the Advanced Encryption Standard or AES-$\{128, 192, 256\}$ (see Definition 1.5) is available in literature [GLRS16] and we will simply use their gate metrics. We denote the AES-k block-cipher encryption function via $\text{AES}^{(k)} : \{0,1\}^{128} \times \{0,1\}^k \longrightarrow \{0,1\}^{128}$ and the function $\text{AES}^{(k)}_{P_i} : \{0,1\}^k \longrightarrow \{0,1\}^{128}$ for $P_i \in \{0,1\}^{128}$ where $P_i \neq P_j$ if $i \neq j$, which maps

$$\text{AES}^{(k)}_{P_i}(x) \mapsto \text{AES}^{(k)}(P_i, x). \tag{2.90}$$

We recall from discussion in Section 1.4 that uniquely specifying a users key requires a set of r known plaintext-ciphertext pairs $((P_1, C_1), \ldots, (P_r, C_R))$ such that we have the promise $\text{AES}(P_i, K) = C_i$ for $i = 1, \ldots, r$ for a single user's key $K \in \{0,1\}^k$. With respect to cryptanalysis of a single user's key, we expect AES-128 and AES-192 we expect to use $r = 2$ plaintext-ciphertext pairs to uniquely specify a user's key with overwhelming probability, whilst for AES-256 we will wish to use $r = 3$ plaintext-ciphertexts. The authors of the original study of the implementation cost of Grover applied to AES define their quantum oracle for the AES key-search problem via the preimage search problem defined by the function $h_{k,r} : \{0,1\}^k \longrightarrow \{0,1\}^{128r}$ and the image $y_h = C_1 \| \ldots \| C_r \in \{0,1\}^{128r}$, where

$$h_{k,r}(x) \mapsto \text{AES}_{P_1}(x) \| \ldots \| \text{AES}_{P_r}(x). \tag{2.91}$$

The quantum evaluation $\mathcal{E}_{h_{k,r}}$ is itself defined by first making $r-1$ copies of the search-register $|x_1 \ldots x_k\rangle$ using $k(r-1)$ $\wedge_1(X)$ gates with a depth of $\lceil \log_2 r \rceil$. In this way the execution of each $\mathcal{E}_{\text{AES}_{P_i}}$ has access to the key we are testing and we do not need to worry about conflicting access privileges to these qubits. The parallel execution of the circuits $\mathcal{E}_{\text{AES}^{(k)}_{P_1}}, \ldots, \mathcal{E}_{\text{AES}^{(k)}_{P_r}}$ is then possible, hence we have that if a single $\mathcal{E}_{\text{AES}^{(k)}_{P_i}}$ circuit requires $k + w_k + 128$ ancilla qubits to implement, we have that $\mathcal{E}_{h_{k,r}}$ requires $r(k + w_k + 128)$ qubits to implement. The quantum bit oracle $\mathcal{O}^{(b)}_{\chi}$ for the AES-k key-search problem using the direct-method from Section 2.4.1 therefore requires $r(k + w_k + 128) + 1$ qubits, $2k(r-1)$ $\wedge_1(X)$ gates, $2r$ applications of $\mathcal{E}_{AES^{(k)}_{P_i}}$, at most $256r$ X gates and a single $\wedge_{128r}(X)$ gate. The costs of the quantum evaluations are given in Table 2.7 below.

	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits
$\mathcal{E}_{\text{AES}^{(128)}_{P_i}}$	$1,380,420$	$1,060,864$	$2,441,284$	$50,688$	$110,799$	984
$\mathcal{E}_{\text{AES}^{(192)}_{P_i}}$	$1,567,296$	$1,204,224$	$2,771,520$	$44,352$	$96,956$	$1,112$
$\mathcal{E}_{\text{AES}^{(256)}_{P_i}}$	$1,956,099$	$1,505,280$	$3,461,379$	$59,904$	$130,929$	$1,336$

Table 2.7: Clifford+T gate costs for the quantum evaluation $\mathcal{E}_{\text{AES}^{(k)}_{P_i}}$ [GLRS16].

It is clear that the key-search problem admits a constraint-based decomposition, so that $\chi_i : \{0,1\}^k \longrightarrow \{0,1\}$ is defined for $i = 1, \ldots, r$ by

$$\chi_i(x) \mapsto \begin{cases} 1 & \text{if } \mathrm{AES}_{P_i}^{(k)}(x) \stackrel{?}{=} C_i \\ 0 & \text{otherwise.} \end{cases} \tag{2.92}$$

The quantum evaluation $\mathcal{E}_{\chi_i}^{(b)}$ for therefore requires $k + w_k + 1$ qubits (where w_k is the number of ancilla qubits required to implement the AES circuit in quantum hardware) and consists of one quantum evaluation $\mathcal{E}_{\mathrm{AES}_{P_i}}$, at most 128 X gates and one $\wedge_{128}(X)$ gate.

We can therefore use it with the methods in Sections 2.4.2 and 2.4.3. As r is small, we could use either the low-qubit approach given in Section 2.4.2, which requires $k + w_k + r + 1$ qubits or the counter-based approach, which requires $k + w_k + \lceil \log_2(r+1) \rceil + 1$ qubits. We will use the counter-based approach, as it is easier to work with quantum evaluations and will demonstrate the benefits of the approach we present in Chapter 6. The cost of the quantum bit oracles using this method can therefore easily be computing using the formula (2.77) from Section 2.4.2 and are given in the table below for several values of r, which again we will later use in Chapter 6.

The cost of the quantum bit oracles using these two approaches are given in Table

$\mathcal{O}_\chi^{(b)}$ for $AES^{(k)}/r$	#Clifford gates	#T gates	#Total gates	T depth	Overall depth	#Qubits	Method
$\mathrm{AES}^{(128)}/2$	5,527,776	4,247,536	9,775,312	105,456	272,034	1,969	Direct evaluation [GLRS16]
$\mathrm{AES}^{(192)}/2$	6,273,264	6,273,264	12,548,544	92,784	204,088	2,225	Direct evaluation [GLRS16]
$\mathrm{AES}^{(256)}/3$	11,745,762	9,037,808	20,783,570	125,936	277,154	4,009	Direct evaluation [GLRS16]
$\mathrm{AES}^{(128)}/2$	8,301,596	6,274,176	14,675,772	313,120	695,314	988	Counter-based (Section 2.4.3)
$\mathrm{AES}^{(192)}/2$	9,422,852	7,234,336	16,657,188	256,104	612,256	1116	Counter-based (Section 2.4.3)
$\mathrm{AES}^{(256)}/3$	19,592,754	15,069,976	34,662,730	616,216	1,360,118	1,340	Counter-based (Section 2.4.3)

Table 2.8: Cost of quantum bit oracles for AES-128 key-search with r plaintext-ciphertexts.

The trade-offs for a concrete problem are now clear. Chapter 6 will focus upon how to remove many of the negative aspects of this trade-off, so that we can use the same number of qubits as the counter-based method, but get a *better* quantum circuit-size than direct evaluation. We will later require the case $r = 1$, which can be performed via direct evaluation.

$\mathcal{O}_\chi^{(b)}$ for $AES^{(k)}/r$	#Clifford gates	#T gates	#Total gates	T depth	Overall depth	#Qubits	Method
$\mathrm{AES}^{(128)}/1$	2,764,120	2,123,760	4,887,880	103,408	266,915	985	Direct evaluation [GLRS16]
$\mathrm{AES}^{(192)}/1$	3,137,744	3,136,624	6,274,368	90,736	198,969	1,113	Direct evaluation [GLRS16]
$\mathrm{AES}^{(256)}/1$	3,915,350	3,012,592	6,927,942	121,840	266,915	1,337	Direct evaluation [GLRS16]

Table 2.9: Cost of quantum bit oracles for AES-128 key-search with r plaintext-ciphertexts.

With the construction methods of quantum oracles explained via the quantum bit oracles for instances of the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ problem and the costs for the key-search problem for the Advanced Encryption Standard, we will proceed to discuss the topic and applications of amplitude amplification. We refer to these costs throughout this thesis, but from now on use only the metrics of circuit-width (#qubits), circuit-size (#total gates) and circuit-depth (overall depth). Given the cost framework, it is plain that the that the other metrics can also be extracted via our cost formula if required. As we are interested in the impact upon the quantum security parameter for cryptosystems, this will be sufficient for our purposes.

Chapter 3

Amplitude amplification

"These go to eleven."

- Nigel Tufnel[Rei84], *(concerning amplification)*

In this chapter we review the well-known theory of *quantum amplitude amplification* [BHMT02], of which quantum search is one particular application. In Section 2.3 we introduced the concept of *quantum phase oracles* and now discuss their use in conjunction with the theory of quantum amplitude amplification in Section 3.1. We conclude by examining the application of amplitude amplification to *quantum search* in Section 3.2, in the form of Grover's algorithm and the associated costs for solving instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ (see Section 1.3) and the key-search problem for AES (see Section 1.4) using the respective quantum oracles in Sections 2.5.2 and 2.6.

3.1 Amplitude amplification

Amplitude amplification [BHMT02] can be considered a generalisation of Grover's quantum search algorithm [Gro96] that utilises two components — an arbitrary quantum algorithm $\mathcal{A}$ that uses no measurements and a quantum phase oracle $\mathcal{O}_\chi$. Whilst we can treat amplitude amplification as a quantum algorithm in of itself, in that we can execute and measure the quantum state, quantum amplitude amplification is better understood as a *quantum sub-routine* which can be used to design quantum algorithms.

In this section we first discuss and prove the generalised amplitude amplification theorem, then give Grover's algorithm as a special case in Theorem 3.10. We will later move to a level of abstraction that requires only a basic knowledge of quantum computing, but will first prove the generalised amplitude amplification theorem in this section before discussing applications, including Grover's algorithm, in Section 3.2. The following simple definition is key to understanding the theorems and results in this thesis.

Definition 3.1 (Success probability of a quantum algorithm [BHMT02])**.**
Let $\mathcal{A}$ be a measurement-free quantum algorithm acting upon n qubits and $\chi : \{0,1\}^n \longrightarrow \{0,1\}$. We say that the *success probability of $\mathcal{A}$ relative to χ* is the probability that measuring the state $\mathcal{A}\,|0^n\rangle$ in the computational basis results in a bitstring $x \in \{0,1\}^n$ such that $\chi(x) = 1$.

In order to discuss the computational advantages that amplitude amplification offers and provide discussion on the costs involved, we first examine a classical method discussed in the original amplitude amplification paper [Høy00] that uses $\mathcal{A}$ to obtain a bitstring $x \in \{0,1\}^n$ such that $\chi(x) = 1$ with high probability.

3.1.1 A classical procedure to achieve success

Suppose we possess the ability to execute an arbitrary measurement-free quantum algorithm $\mathcal{A}$ which possesses a success probability of $a > 0$ relative to $\chi : \{0,1\}^n \longrightarrow \{0,1\}$. By repeatedly creating the quantum state $\mathcal{A}|0^n\rangle$, measuring it and testing the resulting bitstring $x \in \{0,1^n\}$ with a classical evaluation of χ, we will eventually obtain a bitstring such that $\chi(x) = 1$. This can either be done in serial or parallel — we examine the serial case, but note that the procedure is embarrassingly parallel.

The natural question is how many times must we perform the procedure described above to achieve our objective with high probability? As the procedure can be modelled as performing independent Bernoulli trials, the problem can be modelled as finding the expected value of a geometric distribution. As the probability of success upon the k^{th} attempt is

$$\Pr[X = k] = (1-a)^{k-1}a \tag{3.1}$$

we have that the expected number of trials before the success is modelled by the expectation of the geometric distribution

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{k=1}^{\infty}(1-a)^{k-1}a \cdot k && (3.2)\\
&= a \cdot \sum_{k=1}^{\infty}(1-a)^{k-1} \cdot k && (3.3)\\
&= a \cdot \frac{d}{dp}\left(\sum_{k=1}^{\infty}p^k\right) \qquad \text{where } p = 1-a && (3.4)\\
&= a \cdot \frac{d}{dp}\left(p\sum_{k=0}^{\infty}p^k\right) && (3.5)\\
&= a \cdot \frac{d}{dp}\left(\frac{p}{1-p}\right) && (3.6)\\
&= a \cdot \frac{1\cdot(1-p)-(-1)\cdot p}{(1-p)^2} && (3.7)\\
&= a \cdot \frac{a+1-a}{a^2} && (3.8)\\
&= \frac{1}{a}. && (3.9)
\end{aligned}$$

We will therefore require $O(\frac{1}{a})$ executions of $\mathcal{A}$ and the same number of measurements of the quantum state $\mathcal{A}|0^n\rangle$ and *classical evaluations* of $\chi : \{0,1\} \longrightarrow \{0,1\}$. We will return to these costs after we have proven the amplitude amplification theorem (see Theorem 3.6).

We note for now that we have placed special emphasis on the number of executions of $\mathcal{A}$, as opposed to the number of evaluations of $\chi : \{0,1\}^n \longrightarrow \{0,1\}$. We will return to this after amplitude amplification and in the discussion concerning Grover's algorithm (Theorem 3.10).

3.1.2 The amplitude amplification theorem

In this section we state and prove the theory behind quantum amplitude amplification. The proof of quantum amplitude amplification will require will require an understanding of quantum computing up the level provided in Chapter 2. After the quantum amplitude amplitude theorem is proven (see Theorem 3.6) we will move to a higher level of abstraction that simplifies the process of designing quantum search algorithms in this thesis.

The amplitude amplification subroutine can be executed with any quantum phase oracle $\mathcal{O}_\chi$ which is defined by (or defines) a boolean function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and any quantum algorithm $\mathcal{A}$ that uses no measurements. Whilst it has been proven that if $\mathcal{A}$ implements a uniformly randomly chosen unitary operator $U \in \mathbb{C}^{2^n \times 2^n}$, almost any transformation will work for quantum search [Gro98], we will wish to make specific choices for $\mathcal{A}$. Whilst eventually both the quantum phase oracle and chosen quantum algorithm have to be implemented and each have an associated cost, we will ignore the costs in this section and deal only with the mathematical model of quantum computation so that each quantum algorithm acting on n qubits (plus any number of ancillae qubits required for implementing their action efficiently) may be treated as a unitary operator acting upon the complex Hilbert space $\mathbb{C}^{2^n}$.

We will prove the general case of amplitude amplification, which can exploit quantum phase oracles defined by an arbitrary phase angle $0 \leq \phi < 2\pi$, as we lose nothing by this approach and the results required by this thesis are easily obtained by simple substitution of $\phi = \pi$. We first require some basic definitions and terms

Definition 3.2 (Boolean projection operators)**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ be any boolean function and $\mathcal{H}$ be the complex Hilbert space of dimension 2^n with the computational basis $\{|x\rangle : x \in \{0,1\}^n\}$. We define the pair of *boolean projection operators* associated to χ to be

$$P_\chi^{(0)} = \sum_{\substack{x \in \{0,1\}^n \\ \chi(x)=0}} |x\rangle\langle x| \qquad \text{and} \qquad P_\chi^{(1)} = \sum_{\substack{x \in \{0,1\}^n \\ \chi(x)=0}} |x\rangle\langle x| \,. \tag{3.10}$$

This gives us a way method of representing any n qubit quantum state $|\psi\rangle$ in relation to a boolean function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$.

Lemma 3.3 (Boolean partition of Hilbert spaces)**.**
Let $\mathcal{H}$ be the complex Hilbert space of dimension 2^n. All boolean functions $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ partition $\mathcal{H}$ into the direct sum of two subspaces, so that $\mathcal{H} = \mathcal{H}_0 \oplus \mathcal{H}_1$, where

$$\mathcal{H}_i = \mathrm{Span}_{\mathbb{C}}\Big(\big\{ |x\rangle \in \{0,1\}^n : \chi(x) = i \big\} \Big). \tag{3.11}$$

Proof. Any quantum state $|\psi\rangle \in \mathcal{H}$ can be expressed in the form

$$|\psi\rangle = P_\chi^{(0)} |\psi\rangle + P_\chi^{(1)} |\psi\rangle \,, \tag{3.12}$$

where $P_\chi^{(i)} |\psi\rangle \in \mathcal{H}_i$, giving us the decomposition of $\mathcal{H}$ into the direct sum $\mathcal{H} = \mathcal{H}_0 \oplus \mathcal{H}_1$. □

Lemma 3.3 gives us that any quantum state $|\psi\rangle \in \mathcal{H}$ has a projection onto two subspaces defined by the boolean function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$. In the case where χ is a constant function then such a decomposition is clearly trivial, in that either $\mathcal{H}_0$ or $\mathcal{H}_1$ is the trivial Hilbert space. The following proof relies upon the outer-product representation (see Theorem 2.8) of the two-dimensional vector space spanned by these boolean projections.

Theorem 3.4 (Action of the Amplitude Amplification iterator [BHMT02, Høy00])**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ be any boolean function and $0 \leq \phi, \varphi < 2\pi$. Let $\mathcal{H}$ be the complex Hilbert space with computational basis $\left\{ |x\rangle : x \in \{0,1\}^n \right\}$ and $\mathcal{A}$ be any quantum algorithm acting upon $\mathcal{H}$ which uses no measurements. We define the quantum state $|\Psi\rangle = \mathcal{A}|0^n\rangle$ and denote the projections $|\Psi_i\rangle = P_\chi^{(i)}|\Psi\rangle$ for $i = 0, 1$ so that $|\Psi\rangle = |\Psi_0\rangle + |\Psi_1\rangle$.
Let $\mathcal{O}_\chi(\psi)$ and $\mathcal{O}_{0_n}(\phi)$ be quantum phase oracles defined by the functions $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $0_n : \{0,1\}^n \longrightarrow \{0,1\}$ (see Definition 2.20) and define the *generalised amplification operator*

$$Q(\mathcal{A}, \chi, \phi, \varphi) = -\mathcal{A}\mathcal{O}_{0_n}(\phi)\mathcal{A}^{-1}\mathcal{O}_\chi(\varphi). \tag{3.13}$$

Let $a = \langle\Psi_1|\Psi_1\rangle = \sin^2\theta_a$ and $0 < a < 1$ so that $\theta_a \in (0, \frac{\pi}{2})$, then we have that the action of the amplitude iteration operator $Q(\mathcal{A}, \chi, \phi, \varphi)$ on the two-dimensional subspace spanned by the ordered orthonormal basis $\left(\frac{1}{\sqrt{1-a}}|\Psi_0\rangle, \frac{1}{\sqrt{a}}|\Psi_1\rangle\right)$ is given by the matrix

$$M(\mathcal{A}, \chi, \phi, \varphi) = \begin{bmatrix} -\{(1-e^{i\phi})a + e^{i\phi}\} & e^{i\varphi}\{(1-e^{i\phi})\sqrt{a}\sqrt{1-a}\} \\ (1-e^{i\phi})\sqrt{a}\sqrt{1-a} & e^{i\varphi}\{(1-e^{i\phi})a - 1\} \end{bmatrix} \tag{3.14}$$

Proof. It will first be useful to express the operator $Q(\mathcal{A}, \chi, \phi, \varphi)$ in terms of projection operators relative to the ordered orthonormal basis $\left(\frac{1}{\sqrt{1-a}}|\Psi_0\rangle, \frac{1}{\sqrt{a}}|\Psi_1\rangle\right)$. It is clear that

$$\mathcal{O}_\chi(\varphi) = I_n - (1-e^{i\varphi})\frac{1}{a}|\Psi_1\rangle\langle\Psi_1| \tag{3.15}$$

and

$$\begin{aligned} -\mathcal{A}\mathcal{O}_{0_n}(\phi)\mathcal{A}^{-1} &= -\mathcal{A}\Big(I_n - (1-e^{i\phi})|0^n\rangle\langle 0^n|\Big)\mathcal{A}^{-1} \\ &= -\Big(\mathcal{A}I_n\mathcal{A}^{-1} - (1-e^{i\phi})\mathcal{A}|0^n\rangle\langle 0^n|\mathcal{A}^{-1}\Big) \\ &= -\Big(I_n - (1-e^{i\phi})|\Psi\rangle\langle\Psi|\Big) \\ &= \quad (1-e^{i\phi})|\Psi\rangle\langle\Psi| - I_n. \end{aligned} \tag{3.16}$$

From the representation of $Q(\mathcal{A}, \chi, \phi, \varphi)$ by the projection operators given by (3.15) and (3.16), the action of $Q(\mathcal{A}, \chi, \phi, \varphi)$ on the ordered orthonormal basis $\left(\frac{1}{\sqrt{1-a}}|\Psi_0\rangle, \frac{1}{\sqrt{a}}|\Psi_1\rangle\right)$ is

$$\frac{1}{\sqrt{1-a}}|\Psi_0\rangle \mapsto \quad \big\{(1-e^{i\phi})a + e^{i\phi}\big\}\frac{1}{\sqrt{1-a}}|\Psi_0\rangle \;+\; (1-e^{i\phi})\sqrt{a}\sqrt{1-a}\frac{1}{\sqrt{a}}|\Psi_1\rangle, \tag{3.17}$$

$$\frac{1}{\sqrt{a}}|\Psi_1\rangle \mapsto e^{i\varphi}(1-e^{i\phi})\sqrt{a}\sqrt{1-a}\frac{1}{\sqrt{1-a}}|\Psi_0\rangle \;+\; e^{i\varphi}\big\{(1-e^{i\phi})a - 1\big\}\frac{1}{\sqrt{a}}|\Psi_1\rangle. \tag{3.18}$$

From Equations (3.17) and (3.18) it is clear that the two-dimensional subspace spanned by the ordered orthonormal basis $\left(\frac{1}{\sqrt{1-a}}|\Psi_0\rangle, \frac{1}{\sqrt{a}}|\Psi_1\rangle\right)$ is invariant under the action of $Q(\mathcal{A}, \chi, \phi, \varphi)$. The matrix $M(\mathcal{A}, \chi, \phi, \varphi)$ as given in Equation (3.14) is therefore sufficient to represent the action of $Q(\mathcal{A}, \chi, \phi, \varphi)$ upon this subspace and is easily derived from Equations (3.17) and (3.18). □

Theorem 3.5 (Repeated application of $Q(\mathcal{A}, \chi, \pi, \pi)$ [BHT98, BHMT02])**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ be any boolean function and $0 \leq \phi, \varphi < 2\pi$. Let $\mathcal{H}$ be the complex Hilbert space with computational basis $\left\{ |x\rangle : x \in \{0,1\}^n \right\}$ and $\mathcal{A}$ be any quantum algorithm acting upon $\mathcal{H}$ which uses no measurements. We define the quantum state $|\Psi\rangle = \mathcal{A}\,|0^n\rangle$ and denote the projections $|\Psi_i\rangle = P_\chi^{(i)}\,|\Psi\rangle$ for $i = 0, 1$ so that $|\Psi\rangle = |\Psi_0\rangle + |\Psi_1\rangle$.
Let $a = |\langle\Psi_1|\Psi_1\rangle| = \sin^2\theta_a$ and $0 < a < 1$, such that $\theta_a \in (0, \frac{\pi}{2})$. Then for $k \in \mathbb{N}_0$, it holds that

$$Q^k(\mathcal{A}, \chi, \pi, \pi)\mathcal{A}\,|0^n\rangle = \cos\Big((2k+1)\theta_a\Big)\frac{1}{\sqrt{1-a}}\,|\Psi_0\rangle + \sin\Big((2k+1)\theta_a\Big)\frac{1}{\sqrt{a}}\,|\Psi_1\rangle\,. \tag{3.19}$$

In the case $|\langle\Psi_1|\Psi_1\rangle| = a = \sin^2\theta_a \in \{0,1\}$, it holds that $Q^k(\mathcal{A}, \chi, \pi, \pi)\mathcal{A}\,|0^n\rangle = (\mathcal{O}_\chi)^k\,\mathcal{A}\,|0^n\rangle$.

Proof. Theorem (3.4) gives us that in relation to the orthnormal basis $\Big(\frac{1}{\sqrt{1-a}}\,|\Psi_0\rangle\,, \frac{1}{\sqrt{a}}\,|\Psi_1\rangle\Big)$,

$$Q^k(\mathcal{A}, \chi, \pi, \pi)\mathcal{A}\,|0^n\rangle = M^k(\mathcal{A}, \chi, \pi, \pi)\begin{bmatrix}\sqrt{1-a}\\ \sqrt{a}\end{bmatrix}. \tag{3.20}$$

Substitution of $\phi = \varphi = \pi$, $\sqrt{a} = \sin\big(\theta_a\big)$ and $\sqrt{1-a} = \cos\big(\theta_a\big)$ into (3.20) gives us that

$$M^k(\mathcal{A}, \chi, \pi, \pi)\begin{bmatrix}\sqrt{1-a}\\ \sqrt{a}\end{bmatrix} = \begin{bmatrix}\cos(2\theta_a) & -\sin(2\theta_a)\\ \sin(2\theta_a) & \cos(2\theta_a)\end{bmatrix}^k\begin{bmatrix}\cos\big(\theta_a\big)\\ \sin\big(\theta_a\big)\end{bmatrix}. \tag{3.21}$$

We claim that

$$\begin{bmatrix}\cos(2\theta_a) & -\sin(2\theta_a)\\ \sin(2\theta_a) & \cos(2\theta_a)\end{bmatrix}^k\begin{bmatrix}\cos\big(\theta_a\big)\\ \sin\big(\theta_a\big)\end{bmatrix} = \begin{bmatrix}\cos\big((2k+1)\theta_a\big)\\ \sin\big((2k+1)\theta_a\big)\end{bmatrix}. \tag{3.22}$$

and proceed via induction. The base case for $k = 0$ clearly holds by examination of Equation (3.22), hence we assume Equation (3.22) as our inductive hypothesis and the claim follows as

$$\begin{aligned}\begin{bmatrix}\cos(2\theta_a) & -\sin(2\theta_a)\\ \sin(2\theta_a) & \cos(2\theta_a)\end{bmatrix}^{k+1}\begin{bmatrix}\cos\big(\theta_a\big)\\ \sin\big(\theta_a\big)\end{bmatrix} &= \begin{bmatrix}\cos(2\theta_a) & -\sin(2\theta_a)\\ \sin(2\theta_a) & \cos(2\theta_a)\end{bmatrix}\begin{bmatrix}\cos\big((2k+1)\theta_a\big)\\ \sin\big((2k+1)\theta_a\big)\end{bmatrix}\\ &= \begin{bmatrix}\cos\big(2\theta_a\big)\cos\big((2k+1)\theta_a\big) - \sin\big(2\theta_a\big)\sin\big((2k+1)\theta_a\big)\\ \sin\big(2\theta_a\big)\cos\big((2k+1)\theta_a\big) + \cos\big(2\theta_a\big)\sin\big((2k+1)\theta_a\big)\end{bmatrix}\\ &= \begin{bmatrix}\cos\big((2(k+1)+1)\theta_a\big)\\ \sin\big((2(k+1)+1)\theta_a\big)\end{bmatrix}.\end{aligned} \tag{3.23}$$

Equation (3.19) follows trivially from Equations (3.20), (3.21) and (3.22).
That $Q^k(\mathcal{A}, \chi, \pi, \pi)\mathcal{A}\,|0^n\rangle$ is identical to the state $(\mathcal{O}_\chi)^k\,\mathcal{A}\,|0^n\rangle$ when $|\langle\Psi_1|\Psi_1\rangle| = a \in \{0,1\}$ follows from the fact that in this case we have that $|\Psi\rangle = |\Psi_a\rangle$ and so

$$Q(\mathcal{A}, \chi, \pi, \pi)\mathcal{A}\,|0^n\rangle = -\big(I_n - 2\,|\Psi_a\rangle\langle\Psi_a|_n\big)\mathcal{O}_\chi\,|\Psi_a\rangle = \mathcal{O}_\chi(\pi)\,|\Psi_a\rangle \tag{3.24}$$

as $\mathcal{O}_\chi$ only alters the amplitude of the state $|\Psi_a\rangle$, whilst $-\big(I_n - 2\,|\Psi_a\rangle\langle\Psi_a|\,\big)\,|\Psi_a\rangle = |\Psi_a\rangle$. □

Theorems 3.4 and 3.5 give us that we can construct a specific quantum state defined by the quantum algorithm $\mathcal{A}$, boolean function χ and number of times $k \in \mathbb{N}_0$ that we run the amplitude amplification iterator. Whilst it is possible to work with these theorems directly, we will eventually want to measure the quantum state and so it is easier to state them as Theorem 3.6, a statement close to the original formulation of the theorem [BHMT02].

Theorem 3.6 (Amplitude amplification)**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ be any boolean function and $\mathcal{A}$ be a measurement-free quantum algorithm with a success probability of $a > 0$ relative to χ. Let $k \in \mathbb{N}_0$.
Then there exists a quantum algorithm $\mathcal{B}(k)$ with a success probability relative to χ of

$$\sin^2\Big((2k+1)\cdot \arcsin\sqrt{a}\Big) \tag{3.25}$$

and which possesses an execution cost (using the notation in Definition 2.23) of

$$E_{\mathcal{A}} + k\cdot(E_\chi + E_{\bar{n}} + 2E_{\mathcal{A}})\,. \tag{3.26}$$

Proof. This is simply a convenient restatement of the previous two theorems in relation to Definitions 3.1 and 2.23. The success probability is directly obtained from Theorem 3.6 and the computational costs are derived from Theorem 3.6 in combination with the cost of the amplitude amplification operator from Theorem 3.5 using the cost notation from Definition 2.23. □

This notation and methodology make the design and costing of quantum algorithms relatively simple. It is now clear that we can easily assign a cost to any quantum algorithms which solely exploit amplitude amplification and quantum evaluations/quantum bit oracles. We need simply design the quantum algorithm, derive the cost in terms of unitary operators using Equation (3.26) and assign each unitary operator an implementation cost using the results in Section 2.3.

Theorem 3.7 (Quadratic speedup [BBHT96, Gro96, BHMT02])**.**
Let $0 < a \leq 1$ and $k = \left\lfloor \frac{\pi}{4 \cdot \arcsin \sqrt{a}} \right\rfloor$. Then we have that

$$\sin^2\left((2k+1) \cdot \arcsin \sqrt{a}\right) \geq \max\{a, 1-a\}. \tag{3.27}$$

Proof. The optimal $\hat{k} \in \mathbb{R}$ to ensure $\sin^2\left((2\hat{k}+1) \cdot \arcsin \sqrt{a}\right) = 1$ is $\hat{k} = \frac{\pi}{4 \arcsin \sqrt{a}} - \frac{1}{2}$ as

$$\left(2 \cdot \left(\frac{\pi}{4 \cdot \arcsin \sqrt{a}} - \frac{1}{2}\right) + 1\right) \cdot \arcsin \sqrt{a} = \frac{\pi}{2}. \tag{3.28}$$

However, in amplitude amplification an integer $k \in \mathbb{N}_0$ must be chosen as it represents the integer number of times we repeatedly apply the amplitude amplification operator. If we choose $k = \left\lceil \frac{\pi}{4 \cdot \arcsin \sqrt{a}} - \frac{1}{2} \right\rfloor = \left\lfloor \frac{\pi}{4 \cdot \arcsin \sqrt{a}} \right\rfloor$, then we obtain that

$$|\hat{k} - k| \leq \frac{1}{2} \implies \left|(2\hat{k}+1) \cdot \arcsin \sqrt{a} - (2k+1) \cdot \arcsin \sqrt{a}\right| \leq \arcsin \sqrt{a}. \tag{3.29}$$

Substitution of (3.28) then gives us

$$\implies \left|\frac{\pi}{2} - (2k+1) \cdot \arcsin \sqrt{a}\right| \leq \arcsin \sqrt{a} \tag{3.30}$$

and via the fact that $|\sin(|x|)| = |\sin(x)|$ and $\sin\left(\frac{\pi}{2} - x\right) = \cos(x)$, we have that

$$\implies \left|\cos\left((2k+1) \cdot \arcsin \sqrt{a}\right)\right| \leq \left|\sin\left(\arcsin \sqrt{a}\right)\right| = \sqrt{a}, \tag{3.31}$$

which, by squaring both sides and using the fact that $\sin^2(x) = 1 - \cos^2(x)$, provides the bound

$$\implies \sin^2\left((2k+1) \cdot \arcsin \sqrt{a}\right) \geq 1 - a. \tag{3.32}$$

To obtain (3.25), we simply note that if $k = \left\lfloor \frac{\pi}{4 \cdot \arcsin \sqrt{a}} \right\rfloor = 0$, then $\sin^2\left(\arcsin \sqrt{a}\right) = a$. □

Theorem 3.7 gives us k, the optimal number of times to apply the amplitude amplification operator $Q(\mathcal{A}, \chi, \pi, \pi)$ with regards to Theorem 3.6 with the objective of creating a quantum algorithm which succeeds with a probability of at least $1-a$ relative to $\chi : \{0,1\}^n \longrightarrow \{0,1\}$. This gives us both a bound on the probability of success and allows us to quantify the computational resources required to achieve this success probability.

We recall from Section 3.1.1 which describes a classical method to exploit a quantum algorithm $\mathcal{A}$ with a success probability of $a > 0$ relative to $\chi : \{0,1\}^n \longrightarrow \{0,1\}$, that the number of executions of $\mathcal{A}$, measurements of the state and classical evaluations of χ was $O(\frac{1}{a})$. If we assume that $0 < a \ll 1$ then Theorem 3.7 gives us a value of $k = \left\lfloor \frac{\pi}{4 \cdot \arcsin \sqrt{a}} \right\rfloor \approx \frac{\pi}{4} \cdot \frac{1}{\sqrt{a}}$ in conjunction with Theorem 3.6 to succeed with probability close to 1. This is equivalent to $2k+1$ applications of $\mathcal{A}$, k applications of the quantum phase oracles $\mathcal{O}_\chi$, k applications of the quantum phase oracle $\mathcal{O}_{\bar{0}_n}$ and one measurement. Asymptotically, this is $O(\frac{1}{\sqrt{a}})$ applications of the quantum algorithm $\mathcal{A}$, which is clearly an advantage. A direct comparison of costs is not straight forward and we briefly discuss this before describing applications of amplitude amplification.

Whilst an asymptotically amazing result that we need only apply $O(\frac{1}{\sqrt{a}})$ executions of $\mathcal{A}$ and quantum queries to $\mathcal{O}_\chi$ compared to $O(\frac{1}{a})$ executions of $\mathcal{A}$ and classical queries to χ, there are some crucial real-world costs in consider. In the classical procedure we perform multiple independent executions of the quantum algorithm $\mathcal{A}$ and must only protect the quantum state from noise for a relatively short period. Verification that measurement of the quantum state $\mathcal{A}|0^n\rangle$ has produced a bitstring $x \in \{0,1\}^n$ such that $\chi(x) = 1$ is also performed on a classical computer, which will be relatively cheap compared to any such execution on a quantum computer. In comparison, amplitude amplification requires that the quantum state be protected from decoherence for a long-running serial computation and that $\mathcal{O}_\chi$ (often a costly procedure), be executed on the quantum computer.

In this thesis we optimise in the logical quantum circuit model of computation and ignore the requirement for error-correction, but note it for completeness. Our results should be viewed as advantages with respect to existing quantum search procedures and their impact upon cryptographic parameters— whether these optimisation help in any real-world execution of quantum search will depend upon many engineering parameters including the error-correction scheme, hardware tolerance and topology of connections between qubits. These costs may imply that once we factor in real-world costs, quantum search procedures based upon amplitude amplification, including Grover's algorithm and the results in this thesis, may not even begin to offer an advantage over classical search procedures until the search-space is large ($n \approx 55$) [ADMG+16, Ghe17], even with quantum phase oracles which are relatively inexpensive to implement and optimistic assumptions with regards to our ability to perform error-correction.

3.2 Applications of amplitude amplification to quantum search

We now turn to examine applications of amplitude amplification. Grover's quantum search algorithm [Gro96], which we cover in Section 3.2.1 can be seen as a relatively simple application of amplitude amplification [BHMT02] with a fixed choice of quantum algorithm $\mathcal{A} = H^{\otimes n}$ — the Hadamard transform on n qubits (see Definition 3.8). The understanding of the computational benefits it provides is often framed solely in terms of the asymptotic number of queries to the quantum phase oracle $\mathcal{O}_\chi$ compared to the number of classical queries of χ required by the classical search procedure described in Section 3.1.1. This is natural and helpful in understanding the basic costs involved, as the cost of implementing $\mathcal{O}_\chi$ for Grover's algorithm will often dominate the cost of $\mathcal{A}$, but can be misleading if we think of amplitude amplification in these terms.

Our goal in this thesis is to examine modifications of quantum search procedures involving amplitude amplification that can reduce the overall cost of the quantum search procedure when the quantum phase oracle $\mathcal{O}_\chi$ is expensive to implement. Grover's quantum search algorithm will be our benchmark for all comparisons. It will be essential to keep in mind that amplitude amplification offers a quadratic reduction in both the number of quantum queries of $\mathcal{O}_\chi$ versus classical queries of χ *and* the quantum algorithm $\mathcal{A}$. As we proceed to examine applications of amplitude amplification, the first of which is Grover's quantum search algorithm (see Theorem 3.10), we ask the reader to keep this in mind.

In addition to Grover's quantum search algorithm, we will describe an implementation of *exact amplitude amplification* in Section 3.2.3, which has various applications — one of which is the derandomisation of quantum search so that (at least in the logical circuit model of quantum computation), if we know the cardinality of $M_\chi = |\chi^{-1}(1)|$, then we can use the components of amplitude amplification to create a quantum algorithm with a success probability of 1 relative to χ, instead of at least $\max\{a, 1-a\}$.

The nesting of quantum amplitude amplification is a common approach to the design of quantum search algorithms [CGW00], but to the author's knowledge has only been loosely studied as a method to reduce the overall cost of quantum search in the logical quantum circuit model of computation by Kimmel et al. in relation to the *Search with Two Oracles* (*STO*) problem, which we cover and extend to real-world scenarios in Chapter 6. One notable exception is the metaoptimisation of Grover's quantum search algorithm by Arunachalam and de Wolf [AdW17], which examines how a nuanced usage of amplitude amplification can reduce the number of gates *other* than those used in the application of the quantum search oracle. Our methods in Section 5.2 will bear some similarity to their methods.
We first define a basic quantum algorithm, which will be used throughout this thesis.

Definition 3.8 (The Hadamard transform on n qubits)**.**
The *Hadamard transform on n qubits* is the parallel application of n H (Hadamard) gates upon n different qubits. When the qubits are contiguous, we will denote this by $H^{\otimes n}$.

Theorem 3.9 (The uniform superposition)**.**
When applied to the n-qubit state $|0^n\rangle$, the action of the Hadamard transform on n qubits is

$$H^{\otimes n} |0^n\rangle \mapsto \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle . \tag{3.33}$$

$H^{\otimes n} |0^n\rangle$ will be referred to as the *uniform superposition of n qubits* or the *uniform superposition.*

Proof. This can be seen by induction. The case $n = 1$ is simply the Hadamard transform applied to one qubit, which produces the quantum state

$$H |0\rangle \mapsto \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{2^{1/2}} (|0\rangle + |1\rangle) \tag{3.34}$$

by definition of the Hadamard gate (see Section 2.1.6.3), satisfying the case $n = 1$.
Assuming the inductive hypothesis (3.33), we have that

$$H^{\otimes n+1} \left|0^{n+1}\right\rangle \mapsto \frac{1}{2^{1/2}} (|0\rangle + |1\rangle) \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle = \frac{1}{2^{(n+1)/2}} \sum_{x' \in \{0,1\}^{n+1}} \left|x'\right\rangle . \tag{3.35}$$

□

The uniform superposition $H^{\otimes n} |0^n\rangle$ has the property that as the amplitude of each computational basis state $|x\rangle$ is $\alpha_x = \frac{1}{2^{n/2}}$, we have that $|\alpha_x|^n = \frac{1}{2^n}$. Measurement of the uniform superposition therefore gives an equal probability of obtaining any $x \in \{0,1\}^n$. Knowledge of

$M_\chi = |\chi^{-1}(1)|$ then allows us to compute the success probability of $H^{\otimes n}$, which is simply $\frac{M_\chi}{2^n}$. The success probability of $H^{\otimes n}$ is therefore easy to estimate, as it only relies upon knowledge of M_χ. Any other algorithm would require not only knowledge of M_χ, but partial knowledge about the location of the solutions.

3.2.1 Grover's algorithm

Theorem 3.10 (Grover's quantum search algorithm [Gro96, BBHT96])**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $M_\chi = |\chi^{-1}(1)|$ be known. There exists a quantum algorithm with a success probability relative to χ of at least $\max\left\{\frac{M_\chi}{2^n}, 1 - \frac{M_\chi}{2^n}\right\}$ and which costs

$$E_{H^{\otimes n}} + \left\lfloor \frac{\pi}{4 \cdot \arcsin\sqrt{\frac{M_\chi}{2^n}}} \right\rfloor \cdot (E_\chi + E_{\bar{n}} + 2E_{H^{\otimes n}}) \,. \tag{3.36}$$

Proof. If we apply amplitude amplification (see Theorem 3.6) with the quantum algorithm set to be the Hadamard transform on n qubits so that $\mathcal{A} = H^{\otimes n}$, then as $H^{\otimes n}$ has a success probability relative to χ of $a = \frac{M_\chi}{2^n}$, we can use Theorem 3.7 to give us our choice of k and new success probability. □

If we assume that n is large and $M_\chi \ll 2^n$, then via the small angle approximation $\arcsin x \approx x$, we obtain the well-known query complexity of Grover's algorithm of $\left\lfloor \frac{\pi}{4 \arcsin\sqrt{\frac{M_\chi}{2^n}}} \right\rfloor \approx \left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{2^n}{M_\chi}} \right\rfloor$, which gives us that asymptotically we require $O(\sqrt{\frac{2^n}{M_\chi}})$ queries to $\mathcal{O}_\chi$. Zalka proved that Grover's algorithm is optimal [Zal99] in terms of the number of calls to the quantum phase oracle $\mathcal{O}_\chi$ we must make to maximise the success probability for the case $M_\chi = 1$. This was later extended by Fluher to the case where $M_\chi \geq 1$ [Flu17]. Both results are based upon the assumption that the quantum phase oracle is treated as a black-box and that the distribution of elements $x \in \{0,1\}^n$ that satisfy $\chi(x) = 1$ in the domain $\{0,1\}^n$ is uniform. We will later see that if the quantum phase oracle admits specific decompositions, then we can obtain computational gains by using a finer-grained approach to quantum algorithms by using nested applications of amplitude amplification — but that the results of Zalka and Fluher still hold as we will in fact make *more* queries than Grover's algorithm requires, but to cheaper quantum oracles.

In many cases it will hold that E_χ will scale with the size of the search domain. The approximate quantum circuit-size of Grover's algorithm for the case $M_\chi = 1$ will therefore be

$$\approx \frac{\pi}{4} \cdot 2^{n/2} \cdot poly(n) \tag{3.37}$$

and if $n = 100$ and $poly(n) \in O(n^3)$, then we have that Grover's algorithm will require approximately $2^{69.5}$ quantum gates compared to the 2^{50} compared to the lower bound of 2^{50} quantum gates. The aim of this thesis is to examine how structure can be used to reduce this $poly(n)$ overhead imposed by the requirement to implement the quantum phase oracle $\mathcal{O}_\chi$.

Chapter 4 involves only Grover's algorithm whilst Chapters 5 and 6 also involve amplitude amplification. We first briefly consider the concrete cost of solving an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ and the key-search problem for the Advanced Encryption Standard using the oracles given respectively in Tables 2.5 and 2.8.

3.2.1.1 Applying Grover's algorithm to instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$

Using the quantum bit oracle designs in Table 2.5 and formula (3.36) we obtain the following costs for Grover's algorithm applied to instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$. We give the costs relative to the parameters required to forge a Gui signature, as discussed in Section 1.3.7, recalling that to break a security level of $\lambda = 80, 128, 256$ we must respectively solve two serial instances of $\mathcal{MQ}(\mathbb{F}_2, 117, 117)$, $\mathcal{MQ}(\mathbb{F}_2, 209, 209)$ and $\mathcal{MQ}(\mathbb{F}_2, 457, 457)$.

λ	$\mathcal{MQ}(\mathbb{F}_2, n, m)$	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits	Success %	Oracle type
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{79.89}$	$2 \cdot 2^{74.53}$	$2 \cdot 2^{79.97}$	$2 \cdot 2^{74.53}$	$2 \cdot 2^{79.89}$	236	≈ 100.00	Low-memory [SW16]
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{80.88}$	$2 \cdot 2^{76.76}$	$2 \cdot 2^{80.96}$	$2 \cdot 2^{75.57}$	$2 \cdot 2^{80.88}$	126	≈ 100.00	Counter-based [SW16]
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{128.35}$	$2 \cdot 2^{123.38}$	$2 \cdot 2^{128.34}$	$2 \cdot 2^{122.18}$	$2 \cdot 2^{128.35}$	420	≈ 100.00	Low-memory [SW16]
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{129.34}$	$2 \cdot 2^{124.42}$	$2 \cdot 2^{129.39}$	$2 \cdot 2^{123.21}$	$2 \cdot 2^{129.34}$	219	≈ 100.00	Counter-based [SW16]
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{255.69}$	$2 \cdot 2^{249.63}$	$2 \cdot 2^{255.72}$	$2 \cdot 2^{248.42}$	$2 \cdot 2^{255.69}$	916	≈ 100.00	Low-memory [SW16]
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{256.69}$	$2 \cdot 2^{250.65}$	$2 \cdot 2^{256.71}$	$2 \cdot 2^{249.44}$	$2 \cdot 2^{256.69}$	468	≈ 100.00	Counter-based [SW16]

Table 3.1: Quantum resource estimates for solving instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ using Grover.

As it plain from the above table, these security levels are safe (in terms of total number of quantum gates) from a quantum attack via Grover's algorithm using the quantum oracle design principles of Schwabe and Westerbaan discussed in Section 2.5.1. This is unsuprisingly, as the parameters were chosen [PCDY17a, PCDY17b] specifically in relation to an attack Grover's algorithm with Schwabe and Westerbaan's quantum bit oracle.

3.2.2 Applying Grover's algorithm to the key-search problem for AES

We again apply Grover's algorithm to solving the single-target case of the key-search problem for AES-$\{128, 192, 256\}$ as discussed in Section 1.4 and with the quantum bit oracle designs from Table 2.8. We provide quantum resource estimates for both the direct evaluation approach used in the original paper (with a small choice of r as previously discussed in Section 1.4) and the counter-based oracle method from Schwabe and Westerbaan's paper [SW16] introduced in Section 2.4.3.

AES-k/r	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits	Success %	Quantum algorithm/oracle type
128/2	$2^{86.05}$	$2^{85.67}$	$2^{86.87}$	$2^{80.37}$	$2^{81.73}$	1969	≈ 100.00	Grover/direct-evaluation [GLRS16]
128/2	$2^{86.64}$	$2^{86.23}$	$2^{87.45}$	$2^{81.92}$	$2^{83.07}$	988	≈ 100.00	Grover/counter-based [GLRS16]
192/2	$2^{118.23}$	$2^{118.23}$	$2^{119.23}$	$2^{112.2}$	$2^{113.34}$	2225	≈ 100.00	Grover/direct-evaluation [GLRS16]
192/2	$2^{118.82}$	$2^{118.44}$	$2^{119.64}$	$2^{113.63}$	$2^{114.89}$	1340	≈ 100.00	Grover/counter-based [GLRS16]
256/3	$2^{151.14}$	$2^{150.76}$	$2^{151.96}$	$2^{144.64}$	$2^{145.78}$	4009	≈ 100.00	Grover/direct-evaluation [GLRS16]
256/3	$2^{151.88}$	$2^{151.5}$	$2^{152.7}$	$2^{146.89}$	$2^{148.04}$	1340	≈ 100.00	Grover/counter-based [GLRS16]

Table 3.2: Quantum resource estimates for the AES key-search problem via Grover [GLRS16].

3.2.3 Exact amplitude amplification

In this section we describe a variant of *exact amplitude amplification* that will later be referenced.

Theorem 3.11 (Exact amplitude amplification I [BHMT02])**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ be any boolean function and $\mathcal{A}$ be any measurement-free quantum algorithm with a success probability of $a > 0$ relative to χ. Suppose we guess that $a = a_g$. Then there exists a quantum algorithm $\mathcal{C}$ with a success probability relative to χ of

$$\sin^2\left(\left(2\hat{k}_g + 1\right) \cdot \arcsin\sqrt{\frac{a}{a_g} \cdot \sin^2\left(\frac{\pi}{4\hat{k}_g + 2}\right)}\right) \cdot \left(\frac{a_g - a \cdot a_g}{a_g - a \cdot \hat{a}_g}\right) + \frac{a \cdot a_g - a \cdot \hat{a}_g}{a_g - a \cdot \hat{a}_g}, \tag{3.38}$$

where $\hat{a}_g = \sin^2\left(\frac{\pi}{4\hat{k}_g+2}\right)$ and $\hat{k}_g = \left\lceil \frac{\pi}{4 \arcsin\sqrt{a_g}} \right\rceil$. The success probability of $\mathcal{C}$ relative to χ is 1 when $a_g = a$ and $\mathcal{C}$ has a cost of

$$E_{\mathcal{B}} + \hat{k}_g \cdot \left(E_{\hat{\chi}} + E_{\overline{n+1}} + 2E_{\mathcal{B}}\right) \tag{3.39}$$

where $\mathcal{B} = \mathcal{A} \otimes \mathcal{Z}_{\hat{a}_g/a_g}$, we have that $\mathcal{Z}_{\hat{a}_g/a}$ is an arbitrary single-qubit unitary transformation and $\hat{\chi} : \{0,1\}^{n+1} \longrightarrow \{0,1\}$ is defined by $\hat{\chi}(x_1 \ldots x_n x_{n+1}) \mapsto 1$ if and only if $\chi(x_1 \ldots x_n) = x_{n+1} = 1$.

Proof. We first describe one of several methods [BHMT02] of implementing exact amplitude amplification when we have correctly guessed that $a_g = a$. We then derive (3.38) and (3.39).

Guessing correctly. In this case, we have that $a_g = a$. Our goal is to construct a quantum algorithm with a success probability of 1 relative to χ. If $k = \frac{\pi}{4 \arcsin\sqrt{a}} - \frac{1}{2}$ is an integer, then we are already done as Theorem 3.6 implies that the success probability after k iterations is

$$\sin^2\left(\left(2 \cdot \left(\frac{\pi}{4 \arcsin\sqrt{a}} - \frac{1}{2}\right) + 1\right) \cdot \arcsin\sqrt{a}\right) = \sin^2\left(\frac{\pi}{2}\right) = 1. \tag{3.40}$$

If $k \notin \mathbb{N}_0$, then we define $\hat{k} = \left\lceil \frac{\pi}{4 \arcsin\sqrt{a}} \right\rceil$ and will construct a quantum algorithm $\mathcal{B}$ acting upon $n+1$ qubits with success probability $\hat{a} = \sin^2\left(\frac{\pi}{4\hat{k}+2}\right) < a$ relative to $\hat{\chi} : \{0,1\}^n \longrightarrow \{0,1\}$, where

$$\hat{\chi}(x_1 \ldots x_n x_{n+1}) \mapsto \begin{cases} 1 & \text{if } \chi(x_1 \ldots x_n) = 1 \text{ and } x_{n+1} = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{3.41}$$

Theorem 3.6 then implies a quantum algorithm $\mathcal{C}$ with success probability 1 relative to $\hat{\chi}$ as

$$\sin^2\left(\left(2\hat{k} + 1\right) \cdot \arcsin\sqrt{\hat{a}}\right) = \sin^2\left(\left(2\hat{k} + 1\right) \cdot \frac{\pi}{4\hat{k} + 2}\right) = \sin^2\left(\frac{\pi}{2}\right) = 1. \tag{3.42}$$

We now demonstrate how $\mathcal{B}$ can be constructed. We can define the single-qubit unitary operator

$$\mathcal{Z}_{\hat{a}/a} = \begin{bmatrix} \sqrt{1 - \frac{\hat{a}}{a}} & \sqrt{\frac{\hat{a}}{a}} \\ \sqrt{\frac{\hat{a}}{a}} & -\sqrt{1 - \frac{\hat{a}}{a}} \end{bmatrix} \qquad \text{which maps} \qquad \mathcal{Z}_{\hat{a}/a} |0\rangle \mapsto \sqrt{1 - \frac{\hat{a}}{a}} |0\rangle + \sqrt{\frac{\hat{a}}{a}} |1\rangle . \tag{3.43}$$

The quantum algorithm $\mathcal{B} = \mathcal{A} \otimes \mathcal{Z}_{\hat{a}/a}$ has a success probability relative to $\hat{\chi}$ of $\hat{a} = a \cdot \frac{\hat{a}}{a}$. If we consider $\mathcal{C}$ acting upon the n-qubit state $|0^n\rangle$, then $\mathcal{C}$ has a success probability of 1 relative to χ.

Guessing incorrectly. If we perform the algorithm as described on the previous page, but have that $a_g \neq a$ then the definition of $\hat{\chi}$ remains the same, but $\hat{k}_g = \left\lceil \frac{\pi}{4 \arcsin \sqrt{a_g}} \right\rceil$, which defines $\hat{a_g}$ and $\mathcal{Z}_{\hat{a_g}/a_g}$, is derived from a_g instead of a. We therefore have that the quantum algorithm $\mathcal{B} = \mathcal{A} \otimes \mathcal{Z}_{\hat{a_g}/a_g}$ has a success probability of $b = \frac{a}{a_g} \cdot \sin^2\left(\frac{\pi}{4\hat{k}_g+2}\right)$ relative to $\hat{\chi}$ whilst the quantum algorithm $\mathcal{C}$ has a success probability relative to $\hat{\chi}$ of

$$c = \sin^2\left(\left(2\hat{k}_g + 1\right) \cdot \arcsin \sqrt{\frac{a}{a_g} \cdot \sin^2\left(\frac{\pi}{4\hat{k}_g + 2}\right)}\right). \tag{3.44}$$

If again we consider $\mathcal{C}$ as acting upon the n-qubit state $|0^n\rangle$ and treating the additional qubit as an ancilla, then we have that the probability of success of $\mathcal{C}$ relative to χ is the sum of the probabilities that we measure the $n+1$ quantum state and the first n bits of the measurement result in a bitstring $x \in \{0,1\}^n$ such that $\chi(x) = 1$. We must therefore consider the probabilities of the two cases where we measure $x\|0 \in \{0,1\}^{n+1}$ or $x\|1 \in \{0,1\}^{n+1}$ such that $\chi(x) = 1$. The case of measuring $x\|1 \in \{0,1\}^{n+1}$ such that $\chi(x) = 1$ is simply the success probability of $\mathcal{C}$ relative to $\hat{\chi}$ and is therefore c as given by (3.44).

The case where we measure $x\|0 \in \{0,1\}^{n+1}$ such that $\chi(x) = 1$ can be broken down into the product of the probability that we measure a bitstring $x_1 \ldots x_n x_{n+1} \in \{0,1\}^{n+1}$ such that $\hat{\chi}(x_1 \ldots x_n x_{n+1}) = 0$ and the conditional probability that $\chi(x_1 \ldots x_n) = 1$ given that $\hat{\chi}(x_1 \ldots x_n x_{n+1}) = 0$. The probability that $\hat{\chi}(x_1 \ldots x_n x_{n+1}) = 0$ is clearly $1 - c$. The probability that $\chi(x_1 \ldots x_n) = 1$ conditioned on $\hat{\chi}(x_1 \ldots x_n x_{n+1}) = 0$ can be derived by considering the effect of applying the quantum algorithm $\mathcal{B} = \mathcal{A} \otimes \mathcal{Z}_{\hat{a_g}/a_g}$ upon the state $\left|0^{n+1}\right\rangle$.

In this case, the probability that we measure $x_1 \ldots x_n\|0 \in \{0,1\}^{n+1}$ such that $\chi(x_1 \ldots x_n) = 1$ is $a \cdot \left(1 - \frac{\hat{a}_g}{a_g}\right)$, whilst the probability that $\chi(x_1 \ldots x_n) = 0$ is $1 - a = (1-a) \cdot \left(1 - \frac{\hat{a}_g}{a_g}\right) + (1-a) \cdot \frac{\hat{a}_g}{a_g}$. The conditional probability that we measure $x_1 \ldots x_n x_{n+1} \in \{0,1\}^{n+1}$ such that $\chi(x_1 \ldots x_n) = 1$ given that $\hat{\chi}(x_1 \ldots x_n x_{n+1}) = 0$ is therefore

$$\frac{a \cdot \left(1 - \frac{\hat{a}_g}{a_g}\right)}{a \cdot \left(1 - \frac{\hat{a}_g}{a_g}\right) + (1 - a)} = \frac{a \cdot a_g - a \cdot \hat{a}_g}{a_g - a \cdot \hat{a}_g}. \tag{3.45}$$

The success probability of $\mathcal{C}$ relative to χ is then $c + (1-c) \cdot \left(\frac{a \cdot a_g - a \cdot \hat{a}_g}{a_g - a \cdot \hat{a}_g}\right)$, which can be rewritten

$$\sin^2\left(\left(2\hat{k}_g + 1\right) \cdot \arcsin \sqrt{\frac{a}{a_g} \cdot \sin^2\left(\frac{\pi}{4\hat{k}_g + 2}\right)}\right) \cdot \left(\frac{a_g - a \cdot a_g}{a_g - a \cdot \hat{a}_g}\right) + \frac{a \cdot a_g - a \cdot \hat{a}_g}{a_g - a \cdot \hat{a}_g}, \tag{3.46}$$

giving us the success probability (3.38) as stated in Theorem 3.11. It is easily verified that when $a_g = a$, we have that (3.46) is equal to 1. The cost of exact amplitude amplification is therefore

$$E_{\mathcal{B}} + \hat{k}_g \cdot \left(E_{\hat{\chi}} + E_{\overline{n+1}} + 2E_{\mathcal{B}}\right) \quad \text{where} \quad \mathcal{B} = \mathcal{A} \otimes \mathcal{Z}_{\hat{a}_g/a_g} \quad \text{and} \quad \hat{k}_g = \left\lceil \frac{\pi}{4 \arcsin \sqrt{a_g}} \right\rceil. \tag{3.47}$$

□

Exact amplitude amplification can therefore be used to derandomise Grover's quantum search algorithm, as if we precisely know $a_g = a$, then we have a quantum algorithm that terminates with a success probability of 1 with respect to χ. This holds in the mathematical model of quantum computation, where we assume that arbitrary unitary operators can be synthesised, but in the logical quantum circuit model of computation when we have a fixed universal quantum gate set, we have that the correctness is dependent upon the precision to which we can implement the single-qubit quantum gates $\mathcal{Z}_{\hat{a}/a}$. By the Solovay-Kitaev Theorem [KIT96, NC10] single-qubit unitary operators can be efficiently using $O(\log^c(\frac{1}{\epsilon})$ single-qubit gates up to an arbitrary $\epsilon > 0$ level of precision, but we note that in general this implies that in any real-world scenario where we have a fixed universal quantum gate set it will hold that this method will still possess some probability of failure, no matter our choice of ϵ. However, this can be made arbitrarily small.

It is easily seen that the costs of exact amplitude amplification are slightly higher than that required for amplitude amplification as we must perform one more iteration of a slightly more costly amplitude amplification iterator, which requires one more qubit. The costs are therefore comparable, but the advantages dependent upon precise knowledge (or at least an extremely good approximation) of a, the success probability of $\mathcal{A}$ relative to χ, the method by which single-qubit gates are approximated (which determines the constant c) and the use-case. If we can only make an imprecise guess a_g of a, then there is little point in using exact amplitude amplification over amplitude amplification. We will take this approach when we later examine the *Search with Two Oracles* (*STO*) problem [KYYLHH15] in Chapter 6 with respect to real-world problems, where we only know the probability distribution of a. Our main objective in introducing this formulation of exact amplitude amplification will be to demonstrate how the solution to the *STO* problem as put forth by Kimmel et al. [KYYLHH15], which exploits exact amplitude amplification, may fail when we consider real-world scenarios.

Chapter 4

Efficient Neighbourhood Transition Strategies (ENTS)

> *I do not think that all who choose wrong roads perish; but their rescue consists in being put back on the right road. A wrong sum can be put right: but only by going back till you find the error and working it afresh from that point, never by simply going on.*
>
> - C.S. Lewis, *The Great Divorce [LW45]*

In this chapter we examine how the advantages of *multi-target preimage search* can be generalised and exploited to reduce the quantum resources required to execute Grover's algorithm. This chapter introduces the basic principles that we exploit, in some way or another, throughout this thesis. In this section we examine how an *Efficient Neighbour Transition Strategy* (*ENTS*) can be exploited to lower the quantum resources required to execute Grover's algorithm.

4.1 Parallel approaches and the advantages of multiple-targets

Whilst it is well-known that quantum search cannot be parallelised better than by partitioning the search domain and performing parallel searches of these partitions upon different quantum computers [Zal99], this assumes a unit cost for the quantum oracle. We first define some terminology before discussing this further.

Definition 4.1 (Bitmask set of size K)**.**
We say that $Z \subseteq \{0,1\}^n$ is a bitmask set of size $1 \leq K \leq 2^n$ if $|Z| = K$ and $0^n \in Z$.

Definition 4.2 (Bitmask neighbourhood)**.**
Let $Z \subseteq \{0,1\}^n$ be a bitmask set of size K and $x \in \{0,1\}^n$. We say that the set

$$Z_x = \{x \oplus z : z \in Z\} \tag{4.1}$$

is the *bitmask neighbourhood of x of size K*.

In this way, it is plain that as $0^n \in Z$, we have that for all $x \in \{0,1\}^n$ we have that $x \in Z_x$.

4.1.1 The cost of a basic approach to parallel quantum search

If we consider the case of an unstructured search problem defined by $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $M_\chi = |\chi^{-1}(1)| = 1$, then if we possess the quantum phase oracle $\mathcal{O}_\chi$, we can define the new quantum phase oracle $\mathcal{O}_{\chi_Z}$

$$\mathcal{O}_{\chi_Z} |x\rangle \mapsto \begin{cases} -|x\rangle & \text{if } \exists z \in Z : \chi(x \oplus z) = 1 \\ |x\rangle & \text{otherwise.} \end{cases} \tag{4.2}$$

Using the quantum phase oracle $\mathcal{O}_{\chi_Z}$ in conjunction with Grover's algorithm (see Theorem 3.2.1), therefore does not give us the exact answer we are searching for and we must perform a subsequent search procedure afterwards. The initial quantum search with $\mathcal{O}_{\chi_Z}$ as above in (4.2) results in a bitstring $x' \in \{0,1\}^n$ with the property that $x' = x \oplus z$ for some $z \in Z$ — or in other words, $x = x' \oplus z$ for some $z \in Z$. As we precisely know the K bitstrings in Z, we need simply exhaustively evaluate $\chi(x' \oplus z)$ for $z \in Z$ until we find the element that satisfies the original search problem. If the set Z possesses no structure, this may be difficult via quantum search — but if chosen according to some ordering (say $z = 0^{n-k}y \in \{0,1\}^n$ for $|Z_k| = 2^k$), the secondary search can be easily performed via a subsequent quantum search.

Assuming that the quantum phase oracle $\mathcal{O}_\chi$ requires w ancilla qubits to implement, the quantum phase oracle $\mathcal{O}_{\chi_Z}$ requiring $K(n+w)$ qubits can easily be seen to be accomplished in a depth-optimal parallel fashion by using a $\lceil \log_2 K \rceil + 1$ depth circuit involving $n(K-1)$ $\wedge_1(X)$ gates and at most $n(K-1)$ X gates to first compute (for each computational basis state)

$$|x \oplus z_1\rangle |0^w\rangle \ldots |x \oplus z_K\rangle |0^w\rangle \left|0^K\right\rangle \qquad \text{where} \qquad z_1, \ldots, z_K \in Z_x. \tag{4.3}$$

K copies of the quantum phase oracle $\mathcal{O}_\chi$ can then be executed in parallel and the copies of the search space uncomputed. This will implement the quantum phase oracle

$$\mathcal{O}_{\chi'_Z} |x\rangle \mapsto (-1)^k |x\rangle \qquad \text{where} \qquad k = |\{z \in Z_x : \chi(x \oplus z) = 1\}|. \tag{4.4}$$

When $M_\chi = |\chi^{-1}(x)| = 1$, we have that the action of $\mathcal{O}_{\chi'_Z}$ as given by (4.4) is identical to that of $\mathcal{O}_\chi$ as given by $\mathcal{O}_\chi |x\rangle \mapsto (-1)^{\chi(x)} |x\rangle$. In the case where we have that for some $x \in \{0,1\}^n$ we have chosen Z with the property such that $|\{z \in Z : \chi(x \oplus z) = 1\}| = 2k$ for some k, then the approximate quantum phase oracle $\mathcal{O}_{\chi'_Z}$ maps

$$|x\rangle \mapsto (-1)^{2k} |x\rangle = 1^k |x\rangle = |x\rangle , \tag{4.5}$$

thereby having the same effect as leaving the amplitude unchanged. In this case the quantum phase oracle $\mathcal{O}_{\chi'_Z}$ fails to approximate the quantum phase oracle $\mathcal{O}_{\chi_Z}$. In general, so long as $1 \leq M_\chi \ll 2^n$, it is reasonable to assume that this approximation will hold — but we can only guarantee this in the case where $M_\chi = 1$. We assume that $M_\chi = 1$ in the scenarios in this thesis.

This approach requires $K(n+w+1)$ qubits and has an execution cost of $E_{\mathcal{O}_{\chi'_Z}} = K \cdot E_{\mathcal{O}_\chi}$ plus the cost of computing and uncomputing the $K-1$ copies of the register $|x\rangle$ and the bitmasks. By the above discussion this cost has a circuit-depth of $2\lceil \log_2 K \rceil + 2$ and a circuit-size of $4n(K-1)$ Clifford gates. If we assume $E_{\mathcal{O}_{\chi'_Z}}$ dominates all other costs, then we have that whilst the circuit-depth purely benefits from this approach, being (if we assume $K = 2^k$)

$$\approx \frac{\pi}{4} \cdot \frac{2^{n/2}}{2^{k/2}} \cdot E_{\mathcal{O}_\chi^{(b)}} \quad = \frac{\pi}{4} \cdot 2^{(n-k)/2} \cdot E^{(b)}_{\mathcal{O}_\chi} \tag{4.6}$$

we have that the circuit-size is

$$\approx \frac{\pi}{4} \cdot \frac{2^{n/2}}{2^{k/2}} \cdot 2^k E_{\mathcal{O}_\chi^{(b)}} = \frac{\pi}{4} \cdot 2^{(n+k)/2} \cdot E^{(b)}_{\mathcal{O}_\chi}. \tag{4.7}$$

This form of parallelism is therefore useful in terms of reducing the quantum circuit-depth, but will actually increase the total circuit-size of the quantum computation. As Zalka has proved [Zal99], there is essentially no benefit to running Grover's algorithm in parallel on one quantum computer compared to running multiple smaller instances on many separate quantum computers. This was proved relative to the number of calls to the quantum phase oracle $\mathcal{O}_\chi$, but when real-world factors such as the additional circuitry, the difficulty of maintaining protecting the state from noise in larger quantum systems and difficulties in performing massive parallelism is factored in, then running many quantum search routines in parallel with the quantum phase oracle $\mathcal{O}_\chi$ is preferable to executing Grover's algorithm with the quantum phase oracle $\mathcal{O}'_{\chi_Z}$.

However, this is all relative to the quantum bit oracle or quantum phase oracle being treated as a black-box. If we assume some mild structure then we can obtain mild gains in both circuit-depth and circuit-size *without* increasing the number of qubits or even running parallel instances.

4.2 Efficient Neighbourhood Transition Strategies (*ENTS*)

Definition 4.3 (Efficient Neighbourhood Transition Strategy (*ENTS*))**.**
Let $Z = \{z^{(1)}, \ldots, z^{(K)}\} \subseteq \{0,1\}^n$ be a bitmask set of size $1 < K \leq 2^n$ where $z^{(1)} = 0^n$.
We say that $h : \{0,1\}^n \longrightarrow \{0,1\}^m$ admits an *Efficient Neighbourhood Transition Strategy* if there exists a deterministic circuit such that for all $x \in \{0,1\}^n$ and $i = 1, \ldots, K-1$ it holds that evaluating $h(x \oplus z^{(i+1)})$ is cheaper to evaluate if we have already evaluated $h(x \oplus z^{(i)})$.

The folklore differential technique mentioned in Section 1.3.4.2 used in the Fast Exhaustive Search algorithm [BCC+10, BCC+13] for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ is an example of an ENTS as once we have evaluated m equations for the cost of $O(mn^2)$ we can evaluate these equations at another point in the domain that differs by one bitflip for the cost of $O(mn)$. The deterministic circuit is therefore the addition of the bits corresponding to the precomputed $\mathbb{F}_2$-derivatives and the bitmask set is the entire domain $\{0,1\}^n$ as we can enumerate this set via a Gray code, changing only one bit at a time (see Definition 4.8). We give an explicit treatment of its application to quantum search in Section 4.3.3 and provide a method of implementing a Generic ENTS (GENTS) in Section 4.4.2 with application to instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ in Section 4.5.

Employing an ENTS strategy with the single-target preimage search problem as defined by $h : \{0,1\}^n \longrightarrow \{0,1\}^m$, $y_h \in \{0,1\}^m$ and $M_\chi = |h^{-1}(y_h)|$ can therefore be accomplished via the following quantum circuit acting upon $n + m + w$ qubits, where w is enough ancilla qubits to enable computation of both the quantum evaluation $\mathcal{E}_h$ and the unitary operator D_i which for $i = 1, \ldots, K-1$ implements the state transition

$$D_i \left|x \oplus z^{(i)}\right\rangle \left|g(x \oplus z^{(i)})\right\rangle \left|h(x \oplus z^{(i)})\right\rangle \mapsto \left|x \oplus z^{(i+1)}\right\rangle \left|g(x \oplus z^{(i+1)})\right\rangle \left|h(x \oplus z^{(i+1)})\right\rangle . \tag{4.8}$$

Starting from the computational basis state $|x_1 \ldots x_n\rangle |0^w\rangle |0^m\rangle |0\rangle$

1. Execute the quantum evaluation $\mathcal{E}_h$ to obtain the state

$$\left|x \oplus z^{(1)}\right\rangle \left|g(x \oplus z^{(1)})\right\rangle \left|h(x \oplus z^{(1)})\right\rangle |0\rangle . \tag{4.9}$$

2. We then use at most m X gates on the evaluation register to set it to $|1^m\rangle$ if it is equal to y_h, a single $\wedge_m(X)$ gate with the evaluation register as the controls and the last qubit as the target, then at most m X gates again on the equation register to compute

$$\left|x \oplus z^{(1)}\right\rangle \left|g(x \oplus z^{(1)})\right\rangle \left|h(x \oplus z^{(1)})\right\rangle \left|h(x \oplus z^{(1)}) \stackrel{?}{=} y_h\right\rangle . \tag{4.10}$$

3. For $i = 1, \ldots, K-1$:

   (a) Execute the neighbour transition unitary D_i to compute

$$\left|x \oplus z^{(i+1)}\right\rangle \left|g(x \oplus z^{(i+1)})\right\rangle \left|h(x \oplus z^{(i+1)})\right\rangle \left|\bigoplus_{j=1}^{i} (h(x \oplus z^{(j)}) \stackrel{?}{=} y_h)\right\rangle . \tag{4.11}$$

   (b) Apply step 2) using $2m$ X gates and a single $\wedge_m(X)$ gate to compute

$$\left|x \oplus z^{(i+1)}\right\rangle \left|g(x \oplus z^{(i+1)})\right\rangle \left|h(x \oplus z^{(i+1)})\right\rangle \left|\bigoplus_{j=1}^{i+1} (h(x \oplus z^{(j)}) \stackrel{?}{=} y_h)\right\rangle . \tag{4.12}$$

4. Uncompute the quantum evaluation via executing $\mathcal{E}_h^\dagger$ to obtain the state

$$\left|x \oplus z^{(i+1)}\right\rangle |0^w\rangle |0^m\rangle \left|\bigoplus_{j=1}^{K} (h(x \oplus z^{(j)}) \stackrel{?}{=} y_h)\right\rangle . \tag{4.13}$$

5. Use at most n X gates to reset the first register to its original state, leaving the state

$$|x\rangle |0^w\rangle |0^m\rangle \left|\bigoplus_{j=1}^{K} (h(x \oplus z^{(j)}) \stackrel{?}{=} y_h)\right\rangle . \tag{4.14}$$

If $M_h = |h^{-1}(y_h)| = 1$, then this will be correct and if $1 \leq M_h \ll 2^n$ it will be correct with high probability. A description of this circuit for the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ problem is given in Section 4.3.3.

4.2.1 Balancing costs with *ENTS*

We can therefore derive a cost equation for Grover's algorithm if we use this approach, which can be parameterised by K, as we assume that n and m are fixed for the problem

$$E_{\mathcal{G}}(K) = E_{H^{\otimes n}} + \left\lfloor \frac{\pi}{4} \cdot \frac{2^{n/2}}{\sqrt{K}} \right\rfloor \cdot \left(2E_{\mathcal{E}_h} + \sum_{i=1}^{K-1} E_{D_i} + K(E_{\wedge_m(X)} + 2E_{X^{\otimes m}}) + E_{\bar{n}} + 2E_{H^{\otimes n}} \right). \tag{4.15}$$

We note that when $K = 1$, this is simply the special case of using Grover's algorithm with the original quantum bit oracle (constructed using a direct approach as in Section 2.4.1) that we are extending with the ENTS approach. This formula can be minimised towards our chosen cost metric either by numerical simulation (we simply increase K until the costs metric reaches a minimal point) or by relaxing the floor operator and computing the solution to

$$\frac{\mathrm{d}E_{\mathcal{G}}(K, n, m)}{\mathrm{d}K} = 0, \tag{4.16}$$

in order to find the value of $\hat{K} \in \mathbb{R}$ at which $C(K, n, m)$ reaches its minimal point – rounding to the nearest integer to obtain $K = \lfloor \hat{K} \rceil$ will then give us our optimal value of K.

The actual gains using this method will naturally be problem specific and dependent upon the relative cost of $E_{\mathcal{E}_h}$ and E_{D_i}. We give a specific example of this approach and the cost formula applied to using the ENTS strategy to augment Schwabe and Westerbaan's approach to solving instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ in Section 4.3.3. Using this and Table 2.1 for the other costs, we can rewrite the cost formula (excluding the floor function and assuming $D_1 = \cdots = D_{K-1}$)

$$\begin{aligned} E_{\mathcal{G}}(K, n, m) = n &+ \frac{\pi}{4} \cdot 2^{n/2} \cdot K^{-1/2} \cdot \left(2E_{\mathcal{E}_h} + E_{\bar{n}} + 2E_{H^{\otimes n}} - D_i \right) \\ &+ \frac{\pi}{4} \cdot 2^{n/2} \cdot K^{1/2} \cdot \left(E_{D_i} + E_{\wedge_m(X)} + 2E_{X^{\otimes m}} \right) \end{aligned} \tag{4.17}$$

which gives us that

$$\begin{aligned} \frac{\mathrm{d}E_{\mathcal{G}}(K, n, m)}{\mathrm{d}K} = &-\frac{\pi}{8} \cdot 2^{n/2} \cdot K^{-3/2} \cdot \left(2E_{\mathcal{E}_h} + E_{\bar{n}} + 2E_{H^{\otimes n}} - D_i \right) \\ &+ \frac{\pi}{8} \cdot 2^{n/2} \cdot K^{-1/2} \cdot \left(E_{D_i} + E_{\wedge_m(X)} + 2E_{X^{\otimes m}} \right). \end{aligned} \tag{4.18}$$

The optimal value of K can then be derived by solving for K

$$as \frac{\mathrm{d}E_{\mathcal{G}}(K, n, m)}{\mathrm{d}K} = 0 \qquad \iff \qquad K = \left\lceil \frac{2E_{\mathcal{E}_h} + E_{\bar{n}} + 2E_{H^{\otimes n}} - D_i}{E_{D_i} + E_{\wedge_m(X)} + 2E_{X^{\otimes m}}} \right\rfloor. \tag{4.19}$$

Asymptotically, we will therefore have that K is on the order of $O(\frac{E_{\mathcal{E}_h}}{D_i})$ and that the new asymptotic complexity can be obtained via substitution of K into the cost equation. This gives us an algorithm with an asymptotic complexity of (letting $D = D_1 = \cdots = D_{K-1}$)

$$O\left(2^{n/2} \sqrt{D \cdot E_{\mathcal{E}_h}} \right) \tag{4.20}$$

compared with Grover's algorithm used naively with the original oracle, which possesses an asymptotic complexity of $O\left(2^{n/2} E_{\mathcal{E}_h}\right)$. It is plain that if there is a separation of complexity between $E_{\mathcal{E}_h}$ and D (say $E_{\mathcal{E}_h} \in O(n^3)$ and $D \in O(n)$) then we have an asymptotic advantage.

4.3 An example of a problem specific ENTS: The $\mathcal{MQ}$ problem

In this section we demonstrate an example of a problem that admits an Efficient Neighbour Transition Strategy, the $\mathcal{MQ}$ problem over GF(2), provide details of how this can be implemented and analyse the gains. This section may be thought of as a proof of concept for a specific problem that already possesses an efficient neighbour transition strategy.

We recall the classical *Fast Exhaustive Search* (FES) algorithm [BCC+10, BCC+13] for solving instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ that we surveyed in Section 1.3.4.2. Whilst the task of enumerating the solutions of an $\mathcal{MQ}$ system over GF(2) in n variables and m equations requires

$$\mathcal{O}\big(2^n \cdot n^2 m\big) \tag{4.21}$$

operations using a naive exhaustive search (see Section 1.3.4.1), the FES algorithm requires

$$\mathcal{O}\big(2^n \cdot 4\log_2 n\big) \tag{4.22}$$

operations to perform the same task. The key-features it relies upon are that of the $\mathbb{F}_2$-derivative and the *Gray code* — in this section we exploit those features to demonstrate how an *ENTS* strategy can improve the performance of quantum search procedures.

4.3.1 The $\mathbb{F}_2$-derivative

We recall the discussion in Section 1.3.4.2 that the $\mathbb{F}_2$-derivative of a degree d polynomial $f^{(k)}(x_1, \ldots, x_n) \in \mathbb{F}_2[x_1, \ldots, x_n]$ with respective to the variable x_v is of degree $d-1$.

Definition 4.4 (The $\mathbb{F}_2$-derivative [BCC+10])**.**
Let $f \in \mathbb{F}_2[x_1, \ldots, x_n]$. The $\mathbb{F}_2-$derivative of f with respect to x_v for $1 \leq v \leq n$ is defined as

$$\frac{\mathrm{d}f}{\mathrm{d}x_v}(x_1, \ldots, x_n) = f(x_1, \ldots, x_v \oplus 1, \ldots, x_n) \oplus f(x_1, \ldots, x_v, \ldots, x_n). \tag{4.23}$$

Theorem 4.5 (Degree of the $\mathbb{F}_2$-derivative.)**.**
The degree of an $\mathbb{F}_2$-derivative of a degree d polynomial $f \in \mathbb{F}_2[x_1, \ldots, x_n]$ is of degree $d-1$.

Proof. This can be seen as the degree d parts cancel each other out. Explicitly, we have that as any polynomial $f \in \mathbb{F}_2[x_1, \ldots, x_n]$ can be written as the sum and product of x_v and two polynomials $h, g \in \mathbb{F}_2[x_1, \ldots, x_n]$ which do not involve x_v, so that

$$f(x_1, \ldots, x_v, \ldots, x_n) = h(x_1, \ldots, x_n) \oplus x_v \cdot g(x_1, \ldots, x_n) \tag{4.24}$$

where g is of degree d and h is of degree $d-1$, we have that

$$\begin{aligned}
\frac{\mathrm{d}f}{\mathrm{d}x_v}(x_1, \ldots, x_n) &= f(x_1, \ldots, x_v \oplus 1, \ldots, x_n) \oplus f(x_1, \ldots, x_v, \ldots, x_n) && (4.25)\\
&= h(x_1, \ldots, x_n) \oplus (x_v \oplus 1) \cdot g(x_1, \ldots, x_n) \oplus h(x_1, \ldots, x_n) \oplus x_v \cdot g(x_1, \ldots, x_n) && (4.26)\\
&= g(x_1, \ldots, x_n). && (4.27)
\end{aligned}$$

□

For our case, where we are interested in the $\mathbb{F}_2$-derivative of degree-two equations, we therefore have that the $\mathbb{F}_2$-derivative is easily computed.

Theorem 4.6 (The $\mathbb{F}_2$-derivative of quadratic equation)**.**
Let $f \in \mathbb{F}_2[x_1, \ldots, x_n]$ be a degree 2 equation of the form

$$f(x_1, \ldots, x_n) = \sum_{1 \leq i < j \leq n} a_{i,j} x_i x_j + \sum_{i=1}^{n} b_i x_i + c. \tag{4.28}$$

Then the $\mathbb{F}_2$-derivative of f with respect to the variable x_v is

$$\frac{\mathrm{d}f}{\mathrm{d}x_v}(x_1, \ldots, x_n) = b_v + \sum_{i=1}^{v-1} a_{i,v} x_i + \sum_{i=v+1}^{n} a_{v,i} x_i. \tag{4.29}$$

Proof. By inspection. □

The $\mathbb{F}_2$-derivatives of an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ are therefore easily classically computed. Each one requires n bits to store the coefficients and the full set of $\mathbb{F}_2$-derivatives for the instance of MQF requires mn^2 bits of storage for the coefficients.

Theorem 4.7 (The $\mathbb{F}_2$-derivative allows an Efficient Neighbourhood Transition strategy)**.**
Let $f \in \mathbb{F}_2[x_1, \ldots, x_n]$ and $x_1 \ldots x_n \in \{0,1\}^n$ be fixed and suppose we possess the evaluation of $f(x_1, \ldots, x_v, \ldots, x_n)$. Then if we possess the $\mathbb{F}_2$-derivative $\frac{\mathrm{d}f}{\mathrm{d}x_v}(x_1, \ldots, x_n)$, then we can evaluate $f(x_1, \ldots, x_v \oplus 1, \ldots, x_n)$ for the cost of at most $n-1$ additions and 1 bitflip.

Proof. This is simply the folklore-derivative technique discussed in Section 1.3.4.2 and the original Fast Exhaustive Search paper [BCC+10]. Rewriting Equation (4.23) as

$$f(x_1, \ldots, x_v \oplus 1, \ldots, x_n) = \frac{\mathrm{d}f}{\mathrm{d}x_v}(x_1, \ldots, x_v, \ldots, x_n) \oplus f(x_1, \ldots, x_v, \ldots, x_n) \tag{4.30}$$

it is easily seen that as we possess the evaluation $f(x_1, \ldots, x_n)$, we need only evaluate and add $\frac{\mathrm{d}f}{\mathrm{d}x_v}(x_1, \ldots, x_v, \ldots, x_n)$ to obtain $f(x_1, \ldots, x_v \oplus 1, x_n)$. The addition of $\frac{\mathrm{d}f}{\mathrm{d}x_v}(x_1, \ldots, x_v, \ldots, x_n)$ can be performed directly onto the stored value $f(x_1, \ldots, x_v, \ldots, x_n)$, using no additional memory. □

Suppose we possess a method of enumerating $\{0,1\}^n$ via a function $G_n : \{0, \ldots, 2^n - 1\} \mapsto \{0,1\}^n$ for which the Hamming distance between $G_n(i)$ and $G_n(i+1)$ is 1 and that this function has the property that

$$|\{G_n(i) : i = 0, \ldots, 2^n - 1\}| = 2^n \tag{4.31}$$

so that G_n is a bijection, then we can implement a search procedure on an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ via the following procedure.

Assume we have $n+m$ bits of memory. The first n bits will store the *state* of the point in the domain we are currently examining and the last m bits of memory is the *equation register* and will store the evaluation of the m equations. After initialisation of memory:

1. Set the starting point $x_1 \dots x_n \leftarrow G_n(0)$ and store these n bits in the state register.
2. Evaluate each equation on this point, by any method and store these m bits in the equation storage register, so that the register holds $\left(f^{(1)}(x_1, \dots, x_n), \dots, f^{(m)}(x_1, \dots, x_n)\right)$
3. For $i = 1, \dots, 2^{n-1}$:
   (a) Update the state register to $x'_1 \dots x'_n \leftarrow G_n(i)$. Denote the flipped bit as x_v.
   (b) Update the equation register via adding $\frac{\mathrm{d}f^{(k)}}{\mathrm{d}x_v}(x_1, \dots, x_n)$ to each stored evaluation of $f^{(k)}(x_1, \dots, x_n)$ for $k = 1, \dots, m$.
   (c) Store the current state as a solution to the instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ if the equation register is in the state 0^m.

The cost of this method of enumerating the solutions to the instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ is therefore $O(mn^2)$ bit operations to make the initial evaluation of the equations in step 2), whilst the state update function in $3(a)$ can be performed via 1 bitflip and the update of the equation register can be performed using $n-1$ additions and 1 bitflip. The checking of the equation register will require at most m bit operations (though will only require 2 bit comparisons on average by the discussion in Section 1.3.4.2), hence the entire process requires $O(mn^2 + 2^n mn)$ bit operations.

4.3.2 Gray codes

Gray codes [Fra53] fulfil the condition of a successor function $G_n : \{0, \dots, 2^n - 1\} \longrightarrow \{0, 1^n\}$ as stated in Section 4.3.1 and allow the enumeration of the domain $\{0,1\}^n$ of length n binary bitstrings via only $2^n - 1$ bitflips, if we assume that we start from the bitstring 0^n. We use the Gray code as given by the authors of the original Fast Exhaustive Search paper [BCC+10]

Definition 4.8 (Gray code [BCC+10])**.**
Let $\mathrm{G}_n : \{0, \dots, 2^n - 1\} \longrightarrow \{0,1\}^n$ be the Gray code mapping

$$G_n(i) \mapsto i_n \oplus (i_n \gg 1), \tag{4.32}$$

where i_n represents i interpreted as a length n bitstring.

i_n	$G(i_n)$	i_n	$G(i_n)$	i_n	$G(i_n)$	i_n	$G(i_n)$
0000	0000	0100	0110	1000	1100	1100	1010
0001	0001	0101	0111	1001	1101	1101	1011
0010	0011	0110	0101	1010	1111	1110	1001
0011	0010	0111	0100	1011	1110	1111	1000

Table 4.1: Example of the Gray Code upon bitstrings of length 4.

We now proceed to briefly prove that this Gray code has the basic properties that we require.

Theorem 4.9 (G_n is a bijection)**.**
The function $G_n : \{0, \ldots, 2^n - 1\}$ as defined in Definition 4.8 is a bijection.

Proof. If we assume that $i \neq j$ and $G_n(i) = G_n(j)$ then denoting the binary intepretations

$$i_n = i_{n,1} \ldots i_{n,n} \qquad \text{and} \qquad j_n = j_{n,1} \ldots j_{n,n}, \tag{4.33}$$

we have that

$$i_{n,1} \ldots i_{n,n} \oplus 0 i_{n,1} \ldots i_{n,n-1} = G_n(i) = G_n(j) = j_{n,1} \ldots j_{n,n} \oplus 0 j_{n,1} \ldots j_{n,n-1}. \tag{4.34}$$

This implies that $i_{n,1} = j_{n,1}$ as $i_{n,1} \oplus j_{n,1} \oplus 0 \oplus 0 = 0$. As $i_{n,i} = j_{n,i}$ implies that $i_{n,i+1} = j_{n,i+1}$, the result follows from induction. □

Theorem 4.10 ($G_n(i)$ and $G_n(i+1)$ differ by one bit flip.)**.**
For $i = 0, \ldots, 2^n - 2$, the Hamming distance distance between $G_n(i)$ and $G_n(i+1)$ is exactly 1.

Proof. If we consider the sum $G_n(i) \oplus G_n(i+1)$, then (denoting $j = i + 1$), we have that

$$G_n(i) \oplus G_n(i+1) = i_{n,1} \ldots i_{n,n} \oplus (0 i_{n,1} \ldots i_{n,n-1}) \oplus j_{n,1} \ldots j_{n,n} \oplus (0 j_{n,1} \ldots j_{n,n-1}). \tag{4.35}$$

Either $G_n(i)$ will have the property that $i_{n,n} = 1$ or $G_n(i+1)$ will have the property that $j_{n,n} = 1$. If $i_{n,n} = 1$ then we have that $i_{n,1} \ldots i_{n,n} = i_{n,1} \ldots i_{n,n-k-1} 0 1^k$ for some $1 \leq k < n$. This implies that $j_n = i_{n,1} \ldots i_{n,n-k} 1 0^k$, giving us that

$$i_n \oplus j_n = i_{n,1} \ldots i_{n,n-k-1} 0 1^k \oplus i_{n,1} \ldots i_{n,n-k-1} 1 0^k = 0^{n-k-1} 1^{k+1}. \tag{4.36}$$

The other component of the sum $G_n(i) \oplus G_n(i+1)$ is then seen to be

$$(i_n \gg 1) \oplus (j_n \gg 1) = 0 i_{n,1} \ldots i_{n-k-1} 0 1^{k-1} \oplus 0 i_{n,1} \ldots i_{n-k-1} 1 0^{k-1} = 0^{n-k} 1^k. \tag{4.37}$$

This gives us the sum

$$G_n(i) \oplus G_n(i+1) = 0^{n-k-1} 1^{k+1} \oplus 0^{n-k} 1^k = 0^{n-k-1} 1 0^k, \tag{4.38}$$

hence only one bit differs. The proof for $i_{n,n} = 0$ is similar. □

This is all we need to apply our method towards improving Grover's algorithm for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ by an *ENTS* strategy. The actual computation of which bits to flip to enumerate the neighbourhood can be directly computed by a classical control system — as we assume that the bottlenecks in costs will be the quantum resources, not the classical resources, and that the neighbourhood explored by the Gray code will be far smaller than $2^{n/2}$ we can ignore these costs. We note for interest that Lemma 1 of [BCC+10] admits a quicker approach to computing the bit we must flip to obtain $G_n(i+1)$ from $G_n(i)$.

4.3.3 Applying *ENTS* to quantum search for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$

With the $\mathbb{F}_2$-derivative from Definition 4.4 and the Gray code from Definition 4.8, we can now define our *ENTS* strategy for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ using Grover's algorithm. We require any quantum circuit that performs the mapping of computational basis states

$$|x_1 \dots x_n\rangle \, |0\rangle^m \, |0^w\rangle \mapsto |x_1 \dots x_n\rangle \, |f^{(1)}(x_1, \dots, x_n)\rangle \dots |f^{(m)}(x_1, \dots, x_n)\rangle \, |0\rangle \, |0\rangle \,. \tag{4.39}$$

Schwabe and Westerbaan's techniques can perform this action, with the cost of the quantum bit oracles $\mathcal{O}^{(b)}_{f^{(k)}}$ being $O(n^2)$ from Table 2.4.
The full quantum circuit for the *ENTS* approach using the $\mathbb{F}_2$-derivative therefore consists of the following circuit, acting upon each computational basis state $|x_1 \dots x_n\rangle \, |0^m\rangle \, |0\rangle \, |0\rangle$.

1. Execute the m quantum bit oracles $\mathcal{O}^{(b)}_{\tilde{f}^{(1)}}, \dots, \mathcal{O}^{(b)}_{\tilde{f}^{(m)}}$. The computational basis state is now

$$|x_1 \dots x_n\rangle \, |\tilde{f}^{(1)}(x_1, \dots, x_n)\rangle \dots |\tilde{f}^{(m)}(x_1, \dots, x_n)\rangle \, |0\rangle \, |0\rangle \,. \tag{4.40}$$

2. Use a $\wedge_m(X)$ gate with the equation register as controls and the last qubit as a target.

3. Let $v(i)$ be the index of the non-zero bit of $G_n(i-1) \oplus G_n(i)$. For $i = 1, \dots, K-1$:

   (a) We apply an X gate to the $v(i)^{\text{th}}$ qubit of the computational basis state.

   (b) Add the $\mathbb{F}_2$-derivatives $\frac{\mathrm{d}\tilde{f}^{(1)}}{\mathrm{d}x_{v(i)}}(x'_1, \dots, x'_n), \dots, \frac{\mathrm{d}\tilde{f}^{(m)}}{\mathrm{d}x_{v(i)}}(x'_1, \dots, x'_n)$ to their respective positions on the equation register using m X gates and $m(n-1)$ $\wedge_1(X)$ gates.

   (c) Use a $\wedge_m(X)$ gate with the equation register as controls and the last qubit as a target.

4. The equation register is now in the state

$$|x'_1 \dots x'_n\rangle \, |\tilde{f}^{(1)}(x'_1, \dots, x'_n)\rangle \dots |\tilde{f}^{(m)}(x'_1, \dots, x'_n)\rangle \, |0\rangle \, |Z_K(x_1, \dots, x_n)\rangle \,, \tag{4.41}$$

   where $Z_K(x_1, \dots, x_n) = 1$ if and only if $x_1 \dots x_n \oplus G_n(i)$ satisfies the m equations for some $i = 0, \dots, K-1$ (recall that $G_n(0) = 0^n$).
   We apply the quantum bit oracles $\mathcal{O}^{(b)}_{\tilde{f}^{(1)}}, \dots, \mathcal{O}^{(b)}_{\tilde{f}^{(m)}}$ to obtain the computational basis state

$$|x'_1 \dots x'_n\rangle \, |0^m\rangle \, |0\rangle \, |Z_K(x_1, \dots, x_n)\rangle \,. \tag{4.42}$$

5. At most n X gates can then be used on the first n qubits to restore the quantum to

$$|x_1 \dots x_n\rangle \, |0^m\rangle \, |0\rangle \, |Z_K(x_1, \dots, x_n)\rangle \,, \tag{4.43}$$

   with the precise positions we must flip being indicated by the non-zero bits of $G_n(K)$.

For each computational basis state (assuming a single solution to the instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$), we therefore have that $Z_K(x_1, \dots, x_n)$ inverts the phase of $|x\rangle$ if and only if $x \oplus G_n(i)$ for $i = 0, \dots, K-1$ is interpreted as a solution to the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ instance.

The cost of the above quantum bit oracle is therefore higher, but crucially we have introduced new targets into the search-space — this allows us to benefit from a drop in query-complexity. If we denote the unitary D_i as the cost of executing steps $3a$) and $3b$) for the index $i = 1, \ldots, K-1$ from the quantum bit oracle described on the previous page, then we have that the cost function for Grover's algorithm with this approach is therefore

$$E_{H^{\otimes n}} + \left\lfloor \frac{\pi}{4} \cdot \frac{2^{n/2}}{\sqrt{K}} \right\rfloor \cdot \left(2 \sum_{i=1}^{m} E_{\mathcal{O}_{\tilde{f}^{(i)}}} + \sum_{i=1}^{K-1} E_{D_i} + K(E_{\wedge_m(X)} + 2E_{X^{\otimes m}}) + E_{X^{\otimes n}} + E_{\bar{n}} + 2 \cdot E_{H^{\otimes n}} \right). \tag{4.44}$$

Explicitly, we have the cost of these individual components are given by Table 4.2 below, which have respectively been computed using the commentary for the costs in the description of the quantum bit oracle and from Tables 2.1 and 2.4.

	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits
$E_{\mathcal{O}^{(b)}_{\tilde{f}^{(k)}}}$	$n^2 + 11n - 13$	$7n - 7$	$n^2 + 18n - 20$	$3n - 3$	$n^2 + 11n - 16$	$n + 2$
E_{D_i}	$mn + 1$	0	$mn + 1$	0	m	$n + m$
$E_{\wedge_k(X)}$	$24k - 48$	$16k - 16$	$40k - 64$	$16k - 16$	$40k - 64$	$k + 2$
$E_{X^{\otimes n}}/E_{H^{\otimes n}}$	n	0	n	0	1	n
$E_{\bar{n}}$	$26n - 44$	$16n - 16$	$42n - 60$	$16n - 16$	$40n - 62$	$n + 2$

Table 4.2: Clifford+T gate costs for the components used for the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ *ENTS* strategy.

If we then wish to optimise toward the total circuit-size, we obtain the cost equation parameterised by n, m and K

$$C(n, m, K) = n + \left\lfloor \frac{\pi}{4} \cdot \frac{2^{n/2}}{\sqrt{K}} \right\rfloor \cdot \big(2m(n^2 + 18n - 20) + 45n - mn - 61 + K(mn + 42m - 63)\big), \tag{4.45}$$

where we have that the inital cost of evaluating a single point costs $O(mn^2)$, whereas we can test additional points in the search domain for a cost of $O(mn)$ operations each. Assuming n and m are fixed, we can therefore find the minimal quantum circuit-size for this approach either by numerical simulation and varying K or by simply ignoring the floor function and using basic calculus to derive the minimal point, as discussed in Section 4.2.1. Again, it is clear that whilst we are increasing the cost of the Grover iteration, we are reducing the contribution of the query-complexity term to the total cost.

We will optimise towards minimising Equation (4.45), so that we minimise the total quantum circuit-size — the metric we are interested in with regards to the security parameters of the Gui cryptosystem (see Table 1.1). We present our new quantum resource estimates for these parameters in relation to Schwabe and Westerbaan's low-memory quantum bit oracle design and counter-based oracle designs. We note that we cannot use this method with the counter-based-oracle, as the counter-register does not admit (to our knowledge) an *efficient* transition strategy as only the value of whether an equation is satisfied is recorded, not their values.

By relaxing the floor operator and computing

$$\frac{\mathrm{d}C(n,m,K)}{\mathrm{d}K} = -\frac{\pi}{8}\cdot 2^{n/2}K^{-3/2}\cdot\left(2m(n^2+18n-20)+45n-mn-61\right) \tag{4.46}$$

$$+\frac{\pi}{8}\cdot 2^{n/2}K^{-1/2}\cdot\left(mn+42m-63\right) \tag{4.47}$$

and solving for

$$\frac{\mathrm{d}C(n,m,K)}{\mathrm{d}K} = 0 \tag{4.48}$$

we obtain that K optimises the total circuit-size when

$$K = \left\lfloor \frac{2mn^2+35mn-40m+45n-61}{mn+42m-63} \right\rceil \quad . \tag{4.49}$$

We therefore have that K is on the order of $O(n)$, hence the secondary search will be negligible compared to the main quantum search procedure. Substitution of K into the Equation (4.45) gives us that the asymptotic complexity of the search is $O\left(2^{n/2}mn^{3/2}\right)$. This improvement of $O(n^{1/2})$ provides a modest, but concrete impact upon costs and, as Table 4.3 below demonstrates, is enough to break the proposed parameters for the Gui cryptosystem (see Table 1.1). Table 4.3 was derived through numerical simulation to find the optimal value for K by simply computing the cost function for a range of K, which in all cases agrees with Equation (4.49).

λ	$\mathcal{MQ}(\mathbb{F}_2,n,m)$	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits	Success %	K	Oracle type
80	$2\cdot\mathcal{MQ}(\mathbb{F}_2,117,117)$	$2\cdot 2^{77.03}$	$2\cdot 2^{73.43}$	$2\cdot 2^{77.14}$	$2\cdot 2^{73.12}$	$2\cdot 2^{76.43}$	236	≈ 100.00	$199\approx 2^{7.64}$	Low-memory [SW16]
80	$2\cdot\mathcal{MQ}(\mathbb{F}_2,117,117)$	$2\cdot 2^{79.89}$	$2\cdot 2^{74.53}$	$2\cdot 2^{79.97}$	$2\cdot 2^{74.53}$	$2\cdot 2^{79.89}$	236	≈ 100.00	N/A	Low-memory [SW16]
80	$2\cdot\mathcal{MQ}(\mathbb{F}_2,117,117)$	$2\cdot 2^{80.88}$	$2\cdot 2^{76.76}$	$2\cdot 2^{80.96}$	$2\cdot 2^{75.57}$	$2\cdot 2^{80.88}$	126	≈ 100.00	N/A	Counter-based [SW16]
128	$2\cdot\mathcal{MQ}(\mathbb{F}_2,209,209)$	$2\cdot 2^{125.04}$	$2\cdot 2^{120.70}$	$2\cdot 2^{125.11}$	$2\cdot 2^{120.41}$	$2\cdot 2^{124.29}$	420	≈ 100.00	$378\approx 2^{8.56}$	Low-memory [SW16]
128	$2\cdot\mathcal{MQ}(\mathbb{F}_2,209,209)$	$2\cdot 2^{128.35}$	$2\cdot 2^{123.38}$	$2\cdot 2^{128.34}$	$2\cdot 2^{122.18}$	$2\cdot 2^{128.35}$	420	≈ 100.00	N/A	Low-memory [SW16]
128	$2\cdot\mathcal{MQ}(\mathbb{F}_2,209,209)$	$2\cdot 2^{129.34}$	$2\cdot 2^{124.42}$	$2\cdot 2^{129.39}$	$2\cdot 2^{123.21}$	$2\cdot 2^{129.34}$	219	≈ 100.00	N/A	Counter-based [SW16]
256	$2\cdot\mathcal{MQ}(\mathbb{F}_2,457,457)$	$2\cdot 2^{251.79}$	$2\cdot 2^{246.41}$	$2\cdot 2^{251.83}$	$2\cdot 2^{246.13}$	$2\cdot 2^{250.92}$	916	≈ 100.00	$869\approx 2^{9.76}$	Low-memory [SW16]
256	$2\cdot\mathcal{MQ}(\mathbb{F}_2,457,457)$	$2\cdot 2^{255.69}$	$2\cdot 2^{249.63}$	$2\cdot 2^{255.72}$	$2\cdot 2^{248.42}$	$2\cdot 2^{255.69}$	916	≈ 100.00	N/A	Low-memory [SW16]
256	$2\cdot\mathcal{MQ}(\mathbb{F}_2,457,457)$	$2\cdot 2^{256.69}$	$2\cdot 2^{250.65}$	$2\cdot 2^{256.71}$	$2\cdot 2^{249.44}$	$2\cdot 2^{256.69}$	468	≈ 100.00	N/A	Counter-based [SW16]

Table 4.3: Quantum resource estimates for solving $\mathcal{MQ}(\mathbb{F}_2,n,m)$ using Grover with *ENTS*.

From the results in Table 4.3, it is clear that there is no downside (at least with regards to these particular metrics) in using the ENTS method over naively using a quantum bit oracle with Grover's algorithm. We must perform a subsequent search of the space of size K — but as K is on the order of $O(n)$ by the above discussion, this will be a negligible cost and can be assumed to be performed on a classical computer. We can therefore ignore the cost of this subsequent search. We alternatively optimise towards Clifford+T depth, the T gate count or T gate depth.

With this ENTS strategy for instances of $\mathcal{MQ}(\mathbb{F}_2,n,m)$ described and the costs to solve instances of $\mathcal{MQ}(\mathbb{F}_2,n,m)$ provided, it is clear that we have a proof of concept application that the ENTS approach can be used to reduce the costs involved with quantum search for at least one problem. We have additionally broken the quantum-resistant parameters for the Gui cryptosystem by simply optimising the existing approach of Schwabe and Westerbaan by creating a hybrid-quantum classical search algorithm for instances of $\mathcal{MQ}(\mathbb{F}_2,n,m)$.

4.4 Generic Efficient Neighbourhood Transition Strategy (GENTS)

In this section we describe a generic approach to creating an Efficient Neighbourhood Transition Strategy (ENTS) that is based upon how we can structure quantum oracles for better use with Grover's quantum search algorithm (see Theorem 3.10). The process is relatively simple and uses many of the concepts we have already discussed/are well-known in literature. Again, we will wish to balance the query-complexity with the cost of the quantum oracle at the cost of obtaining only partial information concerning the solution. This technique should be viewed through the lens of methods to improve Grover's algorithm via modification of the quantum bit oracle and was developed before the author was familiar with how to structure quantum search via amplitude amplification (see Theorem 3.6). The methods in Chapter 5 extend the basic structure we provide here, demonstrating how we can exploit it separately via both preprocessing and amplitude amplification.

The Generic Efficient Neighbourhood Transition Strategy (*GENTS*) is based upon the fact that we can view any computation as that of a tree, whereby we gradually introduce variables into the computation. This leads to a basic decomposition of a quantum phase oracle.

Definition 4.11 (Tree decomposition of a quantum bit oracle)**.**
Let $\mathcal{O}_\chi^{(b)}$ be a quantum bit oracle defined by the boolean indicator function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ acting upon $n+1$ qubits. We define the *decomposed quantum bit oracle* to be the $n+1$ unitaries

$$U_{\chi_*}, U_{\chi_n}, \ldots, U_{\chi_1} \tag{4.50}$$

acting upon $n+w+1$ qubits, such that for any $x_1 \ldots x_n \in \{0,1\}^n$ and $b \in \{0,1\}$ we have that

$$\left(\mathcal{I}^{\otimes n} \otimes \mathcal{O}_\chi^{(b)}\right) |0^w\rangle |x_1 \ldots x_n\rangle |b\rangle = U_{\chi_1}^\dagger \cdots U_{\chi_n}^\dagger U_{\chi_*} U_{\chi_n} \cdots U_{\chi_1} |0^w\rangle |x_1 \ldots x_n\rangle |b \oplus \chi(x_1 \ldots x_n)\rangle \tag{4.51}$$

where we have $U_{\chi_i} = U'_{\chi_i} \otimes \mathcal{I}^{\otimes n-i+1}$, so that U'_{χ_i} acts upon $w+i$ qubits, with

$$U'_{\chi_i} |g(x_1, \ldots, x_{i-1})\rangle |x_1 \ldots x_i\rangle \mapsto |g(x_1, \ldots, x_i)\rangle |x_1 \ldots x_i\rangle \tag{4.52}$$

where $g(x_1, \ldots, x_i) \in \{0,1\}^w$ is derived only from $x_1, \ldots, x_i$, $g() := 0^w$ and

$$U_{\chi_*} |g(x_1, \ldots, x_n)\rangle |x_1 \ldots x_n\rangle |b\rangle \mapsto |g(x_1, \ldots, x_n)\rangle |x_1 \ldots x_n\rangle |b \oplus \chi(x_1 \ldots x_n)\rangle . \tag{4.53}$$

Each U_{χ_i} should be interpreted as a quantum circuit which takes as input a memory state which has been computed using the variables $x_1, \ldots, x_{i-1}$ and the variable x_i. Each U_{χ_i} should be chosen or designed so that it computes as much as is possible of the quantum bit oracle $U_\chi^{(b)}$ using only the i bits $x_1 \ldots x_i$. The result of this computation is then stored in the working memory register consisting of w qubits. In this way the composed unitary operator

$$U_i := U_{\chi_i} \cdots U_{\chi_1} \tag{4.54}$$

interacts only with the first i qubits of the search space and so the working memory of the output is only entangled with these qubits.

That this decomposition is always possible can be easily demonstrated as we can simply define the trivial decomposition

$$U'_{\chi_i} = \mathcal{I}^{\otimes w+i+1} \tag{4.55}$$

for $i = 1, \ldots, n$ so that $U_n = \mathcal{I}^{\otimes w+n+1}$ and

$$U_{\chi_*} = \mathcal{O}_{\chi}^{(b)}. \tag{4.56}$$

Whether we can exploit this decomposition to achieve computational gains will be dependent upon the individual cost metrics for $E_{U_{\chi_*}}, E_{U_{\chi_n}}, \ldots, E_{U_{\chi_1}}$. Decompositions of this sort are quite naturally common in all areas of computer science and in quantum algorithms. Whilst techniques such as *nested quantum search* [CGW00] and the *quantum backtracking algorithm* [Mon15], these are dependent upon information being revealed at various levels of the tree. Our techniques provide far smaller computational gains than these methods, but make their gains via improving the memory management and balancing the costs of the quantum computation — crucially the only predicate that we use is the unitary U_{χ_*}, which gives us whether the computational basis state $|x_1 \ldots x_n\rangle$ is the answer we are looking for. We first briefly consider the advantages of this decomposition in classical computation.

4.4.1 Exploiting a decomposed classical function

Suppose that we possess the classical analogue of the above decomposition, so that we have a boolean indicator function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ that defines a search problem and that we possess the decomposition

$$\chi(x_1 \ldots x_n) = \chi_*(\chi_n(\chi_{n-1}(\chi_{n-2} \cdots (\chi_2(\chi_1(x_1), x_2), \ldots,)x_n)) \tag{4.57}$$

where

$$\chi_1 : \{0,1\} \longrightarrow \{0,1\}^{w+1} \tag{4.58}$$

$$\chi_i : \{0,1\}^{w+i-1} \times \{0,1\} \longrightarrow \{0,1\}^{w+i} \qquad (\text{for } i = 1, \ldots, n) \tag{4.59}$$

$$\chi_* : \{0,1\}^{w+n} \longrightarrow \{0,1\}. \tag{4.60}$$

If the cost metrics (classical circuit-size or circuit-depth) $E_{\chi_*}, \ldots, E_{\chi_n}, \ldots, E_{\chi_1}$ are defined for $\chi_*, \chi_n, \ldots, \chi_1$, then a simple strategy involves simply computing the intermediate memory state

$$g(x_1, \ldots, x_i) = \chi_i(\chi_{i-1}(\cdots \chi_2(\chi_1(x_1), x_2), \cdots,)x_{i-1}), x_i) \tag{4.61}$$

involving a fixed assignment of k variables corresponding to the bitstring $x_1 \ldots x_k$ and storing this state. We can then perform a simple depth-first search of the implicitly defined search-tree by testing assignments of the remaining $n - k$ variables. Once $\chi(x_1 \ldots x_k \| x_{k+1} \ldots x_n)$ has been evaluated, we can restart the search process from the stored memory state.

This approach can naturally be used in an entirely recursive manner, whereby we store the state $g(x_1, \ldots, x_i) \| x_1 \ldots x_i \in \{0,1\}^{w+i}$ each time after introducing the bit x_i. Using a depth-first search approach whereby we evaluate the 2^n leaves of the implicitly defined perfect binary tree with this technique requires only the storage of at most n intermediate memory states (which including the memory state we are processing). A completely recursive approach therefore requires $nw + \frac{n^2+n)}{2} + 1$ bit of memory and the cost is

$$2^n \cdot E_{\chi_*} + \sum_{i=1}^{n} 2^i \cdot E_{\chi_i}. \tag{4.62}$$

This can be compared to performing a search of the 2^n assignments $x_1 \ldots x_n \in \{0,1\}^n$ without exploiting this decomposition, the cost of which is clearly

$$2^n \cdot \left(\sum_{i=1}^{n} E_{\chi_i} + E_{\chi_*} \right), \tag{4.63}$$

but only requires that we use $n + w + 1$ bit of memory. If we consider a simple problem where $E_{\chi_*} = E_{\chi_i} = C$ for $i = 1, \ldots, n$ then the cost ratio between these two techniques is given by

$$\frac{2^n \cdot (n+1) \cdot C}{(2^n + 2^{n+1} - 2) \cdot C} \approx \frac{n}{3} \tag{4.64}$$

For a problem of size $n = 80$ and assuming a single search target, we therefore have that the reduction in costs will be on the order of $\approx 27 = 2^{4.74}$ and that the advantage will be more extreme if the it holds that the cost of E_{χ_i} decreases as i increases. This is a very simple cost model and ignores many real-world factors, but illustrates the gains that we can make via memory management for evaluating decomposed functions.

The trade-off between memory and circuit-complexity is therefore clear in the case of classical computation. In the case of Grover's algorithm, the implementation of $\mathcal{O}_\chi^{(b)}$ is performed by a *reversible circuit*, hence if we have the tree decomposition of this unitary operator we must execute both U_{χ_i} to compute the successive memory state and $U_{\chi_i}^\dagger$ to uncompute this memory state. If we have this tree decomposition of a quantum bit oracle, then postulate four (see Section 2.1.4) ensures we implicitly keep track of the intermediate memory states at all times anyway. As we are computing these intermediate memory states anyway, we may as well exploit them — or restructure the quantum oracle so that we can exploit them.

This is the reasoning behind the restructuring of Schwabe and Westerbaan's quantum bit oracle for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ that we provide in Section 2.5.2 Equation (2.86) — there are no computational benefits to (2.86) over (2.83) if using a naive evaluation approach, but we can exploit (2.86) using the above technique as the variables x_i are introduced one by one. We explicitly examine how GENTS can exploit this decomposition in Section 4.5.

This will later be adapted into a preprocessing technique to improve the efficiency of attacking instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ in Section 5.1 and a quantum version of this technique in Section 5.2.

4.4.2 The GENTS approach

Given a tree decomposition $U_{\chi_*}, U_{\chi_n}, \ldots, U_{\chi_1}$ of the quantum bit oracle $\mathcal{O}_\chi^{(b)}$ (see Definition 4.11) defined by the boolean indicator function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$, the GENTS approach consists of the following design for the modified quantum bit oracle which acts upon the computational basis state $|0^w\rangle |x_1 \ldots x_n\rangle |b\rangle$, where $b \in \{0,1\}$.

Choose $K \in \mathbb{N}$ to be the size of the bitmask set $Z = \{G_n(i) : i = 0, \ldots, K-1\}$ as defined by the Gray code in Definition 4.8.

1. Execute the composed quantum bit oracle $U_{\chi_*} U_{\chi_n} \cdots U_{\chi_1}$, resulting in

$$U_{\chi_*} U_{\chi_n} \cdots U_{\chi_1} |0^w\rangle |x_1 \ldots x_n\rangle |b\rangle \mapsto |g(x_1, \ldots, x_n)\rangle |x_1 \ldots x_n\rangle |b \oplus \chi(x_1 \ldots x_n)\rangle . \quad (4.65)$$

2. Let $v(i)$ be the index of the non-zero value $G_n(i) \oplus G_n(i-1)$ according to the ordering $x_1 \ldots x_n$ (ie. if $n = 5$, then $G_5(6) \oplus G_5(5) = 00101 \oplus 00111$, so $v(6) = 4$).
   For $i = 1, \ldots, K-1$:
   (a) Execute $U_{\chi_{v(i)}}^\dagger \cdots U_{\chi_n}^\dagger$ (uncomputes the state to the variable we wish to change).
   (b) Apply an X gate to the variable $x_{v(i)}$ (updates the state of the search space).
   (c) Execute $U_{\chi_*} U_{\chi_n} \cdots U_{\chi_{v(i)}}$ (implements the action of $\mathcal{O}_\chi^{(b)}$ using the updated search register for a cheaper cost than a simple parallel approach).

3. Execute the unitary operation $U_{\chi_1}^\dagger U_{\chi_n}^\dagger$ to restore the state to

$$|0^w\rangle \left|x_1' \ldots x_n'\right\rangle \; |b \bigoplus_{i=0}^{K-1} \chi(x_1 \ldots x_n \oplus G_n(i))\rangle \quad (4.66)$$

4. Use at most $\lfloor \log_2(K-1) \rfloor + 1$ X gates to apply the bitmask $G_n(K-1)$ to the search space to obtain the state

$$|0^w\rangle |x_1 \ldots x_n\rangle \; |b \bigoplus_{i=0}^{K-1} \chi(x_1 \ldots x_n \oplus G_n(i))\rangle . \quad (4.67)$$

This allows us to implement a generic approach to the ENTS strategy and we need only design the quantum bit oracles according to the heuristic that as much computation as is possible with the variable x_i must be performed via the circuit for U_{χ_i} before introducing the next variable and doing the same with the variable x_{i+1} and the circuit $U_{\chi_{i+1}}$.

The fact that we have $n+1$ unitaries in our decomposition as opposed to merely n lends itself to a nice interpretation with the preimage search problem defined by $h : \{0,1\}^n \longrightarrow \{0,1\}^m$ and $Y_h \subseteq \{0,1\}^m$. In this scenario the decomposition $U_{\chi_n} \cdots U_{\chi_1}$ corresponds to a decomposition of the quantum evaluation $\mathcal{E}_h$, where $g(x_1, \ldots, x_n) = g'(x_1, \ldots, x_n) \| h(x)$. The unitary U_{χ_*} then corresponds to a comparison circuit which simply tests whether $h(x) \in Y_h$. In the case of the single-target preimage search problem that we are interested in, this has previously been shown to be accomplished by a single $\wedge_m(X)$ gate between two layers of at most m X gates.

4.4.3 On structuring the tree decomposition

In many cases it will prove impossible to create a decomposition that we can easily exploit as all variables must be used immediately in the computation, then a large amount of subsequent computation is performed meaning that $E_{U_{\chi n}}$ dominates the cost and so we suffer from the same circuit-size increase as from the parallel approach discussed in Section 4.1.

One example of this from cryptography is that of the Advanced Encryption Standard [Pub01] discussed in Section 1.4, which owing to how this function is implemented by the repeated iteration of a particular circuit means that as all variables are introduced in the first round of iteration we cannot create a useful decomposition. Potentially interesting sources for making gains using the GENTS approach (and the quantum version, in Section 5.2, which relies upon the same structure) arise from examining the preimage problem applied to cryptographic hash functions, in the context of the Merkle-Damgård extension.

In this scenario, we possess a cryptographic compression function $h : \{0,1\}^{n+c} \longrightarrow \{0,1\}^n$, where $c > 0$ that is difficult to invert. The Merkle-Damgåard construction allows us to extend this construction to create a cryptographic hash function $H : \{0,1\}^* \longrightarrow \{0,1\}^n$ that takes arbitrary size inputs by iteratively hashing the result of a portion of the input with the previous hash, so that if we are hashing the message $m = m_1 \| \ldots \| m_d$, where $m_i \in \{0,1\}^c$, then we can define $z_i = h(z_{i-1} \| m_{i,d})$, where $z_0 := 0^n$. This allows us to define $H(m) = z_d$. In this way, finding a preimage of H has the same structure we wish to exploit — we introduce variables in sets of size c, produce an output memory state, then introduce another set of c variables. Analysis of whether this scenario can be of any practical use is left for future work.

4.5 Applying *GENTS* to instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$

We recall that the definition of a single equation as given by Schwabe and Westerbaan (see Section 2.5.1) was originally

$$f^{(l)}(x_1, \ldots, x_n) = c^{(l)} + \sum_{i=1}^{n} x_i y_i^{(l)} \qquad \text{where} \qquad y_i^{(l)} = b_i^{(l)} + \sum_{j=i+1}^{n} a_{i,j}^{(l)} x_j, \tag{2.83}$$

and that we have changed it in Section 2.86 to the representation

$$f^{(l)}(x_1, \ldots, x_n) = c^{(l)} + \sum_{i=1}^{n} x_i y_i^{(l)} \qquad \text{where} \qquad y_i^{(l)} = b_i^{(l)} + \sum_{j=1}^{i-1} a_{j,i}^{(l)} x_j \tag{2.86}$$

which requires identical resources to evaluate. If we are considering the evaluation of (2.83) using the GENTS approach, then it is clear that in adding $x_1 y_1^{(l)}$ to the equation register involves introducing all n variables immediately, whilst adding $x_n y_n^{(l)}$ to the register involves only the variable x_n. Rewriting this equation as (2.86) simply rewrites the index to take note of this fact.

We cannot apply the approach to the counter-based oracle approach (see Section 2.4.3), as no state information with regards to the equations is kept and using more memory to store these values would negate the purpose of using a counter-based oracle to obtain a low qubit count.

The quantum bit oracle for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ using this approach therefore consists of defining the quantum circuits $U_{\chi_1}, \ldots, U_{\chi_n}$ by (where $b \in \{0, 1\}$)

$$U_{\chi_i} \left|\tilde{f}^{(1)}(x_1, \ldots, x_{i-1})\right\rangle \left|\tilde{f}^{(m)}(x_1, \ldots, x_{i-1})\right\rangle |0\rangle \left|x_1 \ldots x_n\right\rangle |b\rangle \tag{4.68}$$

$$\mapsto \left|\tilde{f}^{(1)}(x_1, \ldots, x_i)\right\rangle \left|\tilde{f}^{(m)}(x_1, \ldots, x_i)\right\rangle |0\rangle \left|x_1 \ldots x_n\right\rangle |b\rangle \tag{4.69}$$

where we define the partial sum

$$\tilde{f}^{(l)}(x_1, \ldots, x_i) = \sum_{j=1}^{i} x_i y_i^{(l)} \tag{4.70}$$

and U_{χ_*} is a single $\wedge_m(X)$ gate with the equation registers as the controls and $|b\rangle$ as the target.

Setting $K = 2^k$ for $0 \leq k \leq n$, the cost formula for the total circuit-size is the cost to evaluate m polynomials over $\mathbb{F}_2$ of degree two involving $n - k$ variables, plus the cost of exploring the 2^k leaf nodes from this point and the cost of the diffusion step. Using the cost of the quantum evaluations from Table 2.4 and the cost of the other components from Table 2.1, this gives us the following cost formula for using Grover's algorithm to solve an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ with $n + m + 2$ qubits using GENTS to augment Schwabe and Westerbaan's approach

$$\begin{aligned} n + \left\lfloor \frac{\pi}{4} \cdot 2^{(n-k)/2} \right\rfloor \cdot \Big(& 2m((n-k)^2 + 18(n-k) - 20) + 44n - 60 \\ & + 2^k(40n - 64) + 2m \sum_{i=1}^{k} 2^i(17 + 2(n - k + i)) \Big). \end{aligned} \tag{4.71}$$

If instead we use a hybrid of this approach and the quantum oracle using $n + m(n-1) + 1$ qubits from Section 2.5.3, then another approach presents itself. In terms of quantum circuit-size, there is little advantage in using this quantum bit oracle, but the quantum evaluation saves roughly a factor of two owing to the fact that we do not have to uncompute $y_i^{(l)}$ before computing $y_{i+1}^{(l)}$. It offers a further advantage in that by using $m(n-2)$ registers to store the $y_i^{(l)}$, we need not uncompute them and can add these bits to these registers as and when they become available. In this way, we have a quantum oracle parameterised by $K = 2^k$, where $0 \leq k \leq n$ and using $n + (k+1)m + 1$ qubits:

Let $U_{\chi_{n-k}} \cdots U_{\chi_1}$ consist of the following procedure:

- Compute the partial sums $\left|f^{(1)}(x_1, \ldots, x_{n-k})\right\rangle \ldots \left|f^{(m)}(x_1, \ldots, x_{n-k})\right\rangle$ on the equation registers using bit oracles to leave the mk temporary storage registers in the state $\left|0^{mk}\right\rangle$.
- In mk registers, compute and store the components of $y_{n-k+1}^{(1)}, \ldots, y_n^{(1)}, \ldots, y_{n-k+1}^{(m)}, \ldots, y_n^{(m)}$ involving only $x_1, \ldots, x_{n-k}$ for a total cost of $mk(n-k-1)$ $\wedge_1(X)$ gates and mk X gates.

Then define U_i for $i = n - k + 1, \ldots, n$ as the following procedure:

- Add (if required) x_i to the $m(n-i+1)$ stored values $y_i^{(l)}, \ldots, y_n^{(l)}$ and add the m values of $x_i y_i^{(l)}$ to the partial sums. This requires m $\wedge_2(X)$ gates and $m(n-i+1)$ $\wedge_1(X)$ gates.

U_{χ_*} is a $\wedge_m(X)$ gate, the controls being the equation registers and the output bit as the target.

This gives us a structuring of the U_{χ_i} whereby E_{χ_i} decreases with i, instead of increasing. Taken all together we have the total circuit-size of Grover's algorithm used in conjunction with this oracle is given by the formula

$$n + \left\lfloor \frac{\pi}{4} \cdot 2^{(n-k)/2} \right\rfloor \cdot \Big(2m((n-k)^2 + 18(n-k) - 20) + 44n - 60 + 2mk(n-k) + 2^k(40m - 64) + 2m \sum_{i=1}^{k} 2^i(17 + k + 1 - i) \Big) \tag{4.72}$$

4.5.1 Asymptotic and concrete results

If we take the derivative of the higher order terms of Equations (4.71) and (4.72) with respect to k, then we can again find the optimal value of k. This turns out to be on the order of $O\left(\log_2(n)\right)$ for the low-qubit approach (Equation (4.71)) and $O\left(\log_2(n^2)\right)$ for the high-qubit approach (Equation (4.72)). After substitution, we find that the asymptotic complexity of the GENTS approach with the low-qubit approach is $O\left(2^{n/2}mn^{3/2}\right)$ and $O\left(2^{n/2}mn\right)$ for the high-memory approach — by using additional qubits we have further improved the quantum circuit-size. To find the exact optimal value of k, we again use the approach of simply using the cost formula given by Equation (4.72) and increasing k until we find the minimal point. Using this approach we achieve the results in Table 4.4 below.

λ	$\mathcal{MQ}(\mathbb{F}_2, n, m)$	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits	Success %	k	Oracle type
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{78.35}$	$2 \cdot 2^{74.05}$	$2 \cdot 2^{78.43}$	$2 \cdot 2^{73.12}$	$2 \cdot 2^{77.23}$	236	≈ 100.00	5	Low-memory [SW16]/GENTS
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{75.79}$	$2 \cdot 2^{75.07}$	$2 \cdot 2^{76.48}$	$2 \cdot 2^{74.38}$	$2 \cdot 2^{76.05}$	1288	≈ 100.00	9	High-memory [SW16]/GENTS
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{79.89}$	$2 \cdot 2^{74.53}$	$2 \cdot 2^{79.97}$	$2 \cdot 2^{74.53}$	$2 \cdot 2^{79.89}$	236	≈ 100.00	N/A	Low-memory [SW16]
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{80.88}$	$2 \cdot 2^{76.76}$	$2 \cdot 2^{80.96}$	$2 \cdot 2^{75.57}$	$2 \cdot 2^{80.88}$	126	≈ 100.00	N/A	Counter-based [SW16]
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{126.44}$	$2 \cdot 2^{121.32}$	$2 \cdot 2^{126.47}$	$2 \cdot 2^{120.41}$	$2 \cdot 2^{125.27}$	420	≈ 100.00	6	Low-memory [SW16]/GENTS
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{123.63}$	$2 \cdot 2^{122.4}$	$2 \cdot 2^{124.14}$	$2 \cdot 2^{121.72}$	$2 \cdot 2^{123.39}$	2509	≈ 100.00	10	High-memory [SW16]/GENTS
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{128.35}$	$2 \cdot 2^{123.38}$	$2 \cdot 2^{128.34}$	$2 \cdot 2^{122.18}$	$2 \cdot 2^{128.35}$	420	≈ 100.00	N/A	High-memory [SW16]
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{123.39}$	$2 \cdot 2^{122.40}$	$2 \cdot 2^{124.14}$	$2 \cdot 2^{121.72}$	$2 \cdot 2^{123.38}$	2509	≈ 100.00	N/A	Counter-based [SW16]
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{253.25}$	$2 \cdot 2^{247.03}$	$2 \cdot 2^{253.26}$	$2 \cdot 2^{246.10}$	$2 \cdot 2^{252.26}$	916	≈ 100.00	7	Low-memory [SW16]/GENTS
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{249.71}$	$2 \cdot 2^{248.97}$	$2 \cdot 2^{250.39}$	$2 \cdot 2^{248.31}$	$2 \cdot 2^{249.92}$	6856	≈ 100.00	13	Low-memory [SW16]
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{255.69}$	$2 \cdot 2^{249.63}$	$2 \cdot 2^{255.72}$	$2 \cdot 2^{248.42}$	$2 \cdot 2^{255.69}$	916	≈ 100.00	N/A	Low-memory [SW16]
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{256.69}$	$2 \cdot 2^{250.65}$	$2 \cdot 2^{256.71}$	$2 \cdot 2^{249.44}$	$2 \cdot 2^{256.69}$	468	≈ 100.00	N/A	Counter-based [SW16]

Table 4.4: Quantum resource estimates for employing GENTS to solve $\mathcal{MQ}(\mathbb{F}_2, n, m)$.

The GENTS approach, like the ENTS approach can be used to break the parameters for the Gui cryptosystem. The low-memory version is not as efficient, but the high-memory version is superior — at least in terms of total gate count. Unfortunately by the nature of GENTS being a generic approach, we must employ the $\wedge_2(X)$ gate when we wish to repeatedly add the values of $|y_i^{(l)}\rangle$ to the equation registers — the ENTS approach bypasses this as we only require the addition of a linear equation and can therefore use $\wedge_1(X)$ gates. The GENTS approach would therefore seem to be useful, but we stress is only a starting point for the improvements in Chapter 5, where we examine how preprocessing can be applied to the GENTS approach to derive a method that is superior to the ENTS strategy for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ and examine how amplitude amplification can be used, as opposed to Grover's algorithm.

Chapter 5

PARENTS and quantum nesting

"Said the mother Tern to her baby Tern
"Would you like a brother?"
Said the baby Tern to the mother Tern
"Yes. One good Tern deserves another."

- Spike Milligan, *[MG03]*

In this chapter we examine several different approaches to exploiting the GENTS approach. In Section 5.1 we meet the PARENTS approach, demonstrating how we can use classical preprocessing to implement the action of the GENTS quantum circuit for a smaller cost than in Section 5.1 when applying the method to instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$. In Section 5.2 we examine the nested application of amplitude amplification in order to improve the efficiency of the search upon the last set of variables. In both sections we apply the procedures to instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ to examine their concrete effect upon parameter choices for Gui.

5.1 Applying preprocessing to the GENTS approach

The following is adapted from work published and presented at the *International Workshop on the Arithmetic of Finite Fields* (Bergen) in 2018 and set in the context of the GENTS approach discussed in Chapter 4. A natural question is how we can reduce the cost of Grover's algorithm in conjunction with the GENTS approach by using classical preprocessing to reduce the cost of implementing the tree-decomposition of the quantum bit oracle. In this section we introduce an improvement to the GENTS approach based on the idea that we can perform preprocessing on the search-tree in order to obtain computational gains with respect to the GENTS approach.

5.1.1 Preprocessing and intermediate memory states

We have defined the tree decomposition of the quantum bit oracle (see Definition 4.11) $\mathcal{O}_{\chi}^{(b)}$ defined by the boolean function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ so that $U_{\chi_1}, \ldots, U_{\chi_n}, U_{\chi*}$ are $n+1$ unitary operators representing quantum circuits acting upon $w+n+1$ qubits, where $U_{\chi_i} = U'_{\chi_i} \otimes \mathcal{I}^{\otimes n-i+1}$ is defined by

$$U'_{\chi_i} \left|g(x_1, \ldots, x_{x_{i-1}})\right\rangle |x_1 \ldots x_i\rangle \mapsto |g(x_1, \ldots, x_{x_i})\rangle |x_1 \ldots x_i\rangle \tag{5.1}$$

so that U'_{χ_i} acts upon $w+i$ qubits and $U_{\chi*}$ is defined as

$$U_{\chi*} |g(x_1, \ldots, x_n)\rangle |x_1 \ldots x_n\rangle |b\rangle \mapsto |g(x_1, \ldots, x_n)\rangle |x_1 \ldots x_n\rangle |b \oplus \chi(x_1 \ldots x_n)\rangle . \tag{5.2}$$

We recall that the bitstring $g(x_1, \ldots, x_i) \in \{0,1\}^w$ represents the state of the memory after introducing the bit x_i into the computation. It will hold that for some $0 < k < n$ that the final memory state can further be expressed by the decomposition

$$g(x_1, \ldots, x_n) = g_1(x_1, \ldots, x_{n-k}) \oplus g_2(x_1, \ldots, x_{n-k}, x_{n-k+1}, \ldots, x_n). \tag{5.3}$$

where again $g_1(x_1, \ldots, x_{n-k})$ depends only upon the bits $x_1, \ldots, x_{n-k}$ whilst the intermediate memory state $g_2(x_1, \ldots, x_{n-k}, x_{n-k+1}, \ldots, x_n)$ is dependent upon all the variables. In the case where $k = 0$ or $k = n$, this is clearly a trivial decomposition where either $g_1() = 0^w$ or we can set $g_2(x_1, \ldots, x_n) = 0^w$, but will have an interpretation with respect to our approach. We will wish $0 < k < n$ for our approach to be useful and to express $g_2(x_1, \ldots, x_{n-k}, x_{n-k+1}, \ldots, x_n)$ algebraically, so that each bit is a polynomial over $\mathbb{F}_2[x_1, \ldots, x_{n-k}, x_{n-k+1}, \ldots, x_n]$ of any degree.

We can then perform a preprocessing step on a classical computer, exhaustively computing the list of w polynomials over $\mathbb{F}_2[x_1, \ldots, x_{n-k}]$ obtained by substituting the variables $x_{n-k+1}, \ldots, x_n$ in $g_2(x_1, \ldots, x_{n-k}, x_{n-k+1}, \ldots, x_n)$ for the 2^k bits $z_{n-k+1} \ldots z_n \in \{0,1\}^k$, giving us the list

$$S_k = \Big[g_2(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n) : z_{n-k+1}, \ldots, z_n \in \{0,1\}\Big], \tag{5.4}$$

which we index from $i = 0, \ldots, 2^k - 1$ and use the notation $S_k[i]$ for the i^{th} element of this list. We can then use the Preprocessed Algebraic Replacement Efficient Neighbourhood Transition Strategy (PARENTS) to compute the action of the quantum bit oracle, where as with ENTS and GENTS, the technique will be correct if the search problem defined by $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ has the property that $M_\chi = |\chi^{-1}(1)| = 1$. In the case where $M_\chi > 1$, our technique will still be correct if there do not exist $x, y \in \chi^{-1}(1)$ whose first $n-k$ bits are identical.

After creating the preprocessed list of equations S_k, we define $U_{g_2}^{(i)}$ for $i = 0, \ldots, 2^k$ by their actions (where $d \in \{0,1\}^w$ and $b \in \{0,1\}$)

$$U_{g_2}^{(0)} \ |d\rangle |x_1 \ldots x_{n-k}\rangle |b\rangle \mapsto |d \oplus S_k[0]\rangle |x_1 \ldots x_{n-k}\rangle |b\rangle \tag{5.5}$$

$$U_{g_2}^{(i)} \ |d\rangle |x_1 \ldots x_{n-k}\rangle |b\rangle \mapsto |d \oplus S_k[i-1] \oplus S[i]\rangle |x_1 \ldots x_{n-k}\rangle |b\rangle \qquad \text{for } i = 1, \ldots 2^k - 1 \tag{5.6}$$

$$U_{g_2}^{(2^k)} \ |d\rangle |x_1 \ldots x_{n-k}\rangle |b\rangle \mapsto |d \oplus S_k[2^k - 1]\rangle |x_1 \ldots x_{n-k}\rangle |b\rangle . \tag{5.7}$$

The sum $S_k[i-1] \oplus S_k[i]$ can also be classically preprocessed. In this way, we have that applying $U_{\chi_{n-k}} \cdots U_{\chi_1}$ followed by $U_{g_2}^{(0)}$ results in the memory storage register holding the memory state

$$|g_1(x_1, \ldots, x_{n-k}) \oplus g_2(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n)\rangle = |g(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n)\rangle \tag{5.8}$$

for the $z_{n-k+1} \ldots z_n$ used to create $S_k[0]$. The successive action of $U_{g_2}^{(i)}$ for $i = 1, \ldots, 2^{k-1}$ will add the difference $S_k[i-1] \oplus S_k[i]$ to the storage register, resulting in the memory storage register holding (5.8) for the $z_{n-k+1} \ldots z_n$ used to create $S_k[i]$. The unitary $U_{g_2}^{(2^k)}$ resets the state to

$$|g_1(x_1, \ldots, x_{n-k})\rangle . \tag{5.9}$$

As we have cycled through the intermediate memory states described by (5.8) for a fixed $x_1 \ldots x_{n-k} \in \{0,1\}^{n-k}$ and the 2^k possible values for $z_{n-k+1} \ldots z_n \in \{0,1\}^k$, we have that the action of U_{χ_*} after applying $U_{g_2}^{(i)}$ for $i = 0, \ldots, 2^{k-1}$ will be as described in (4.60). Finally, the intermediate memory state (5.9) can be uncomputed by the action of $U_{\chi_1}^\dagger \cdots U_{\chi_{n-k}}^\dagger$. Formally, the quantum bit oracle acts upon $w + n - k + 1$ qubit state $|0^w\rangle |x_1 \ldots x_{n-k}\rangle |b\rangle$ and consists of:

1. Apply the unitary operators $U_{\chi_{n-k}} \cdots U_{\chi_1}$ to obtain $|g_1(x_1, \ldots, x_{n-k})\rangle |x_1 \ldots x_{n-k}\rangle |b\rangle$.
2. For $i = 0, \ldots, 2^k - 1$
   (a) Apply $U_{g_2}^{(i)}$ to obtain

$$|g(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n)\rangle |x_1 \ldots x_{n-k}\rangle \ |b \bigoplus_{j=0}^{i-1} \chi(x_1 \ldots x_{n-k} z_{n-k+1} \ldots z_n)\rangle . \tag{5.10}$$

   (b) Apply U_{χ_*} to obtain

$$|g(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n)\rangle |x_1 \ldots x_{n-k}\rangle \ |b \bigoplus_{j=0}^{i} \chi(x_1 \ldots x_{n-k} z_{n-k+1} \ldots z_n)\rangle . \tag{5.11}$$

3. Apply the unitary operator $U_{g_2}^{(2^k)}$ to obtain

$$|g_1(x_1, \ldots, x_{n-k})\rangle |x_1 \ldots x_{n-k}\rangle \ |b \bigoplus_{j=0}^{i} \chi(x_1 \ldots x_{n-k} z_{n-k+1} \ldots z_n)\rangle . \tag{5.12}$$

4. Apply the sequence of unitary operators $U_{\chi_1}^\dagger \cdots U_{\chi_{n-k}}^\dagger$ to obtain

$$|0^w\rangle |x_1 \ldots x_{n-k}\rangle \ |b \bigoplus_{j=0}^{2^k-1} \chi(x_1 \ldots x_{n-k} z_{n-k+1} \ldots z_n)\rangle . \tag{5.13}$$

We give an explicit example of how we can adapt this method to improve Schwabe and Westerbaan's quantum bit oracle [SW16] (see Section 2.5.1), which was the original work published [Pri18] at WAIFI 2018 by the author.

5.1.2 Introducing the PARENTS to your instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$

In the context of instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$, the PARENTS approach has the following interpretation. We use PARENTS in conjunction with the low-memory quantum oracle design for the $\mathcal{MQ}$ problem (see Section 2.4.2) and the cost for the quantum bit oracles $\mathcal{O}_{\tilde{f}^{(k)}}$ for individual equation given by Table 2.5. As before, we cannot use this approach with the counter-based quantum oracle design (see Section 2.4.3). The memory register uses $w = m + 1$ qubits, where 1 qubit is used for temporary storage for computation and m qubits act as an equation register which will store the evaluated equations

$$|\tilde{f}^{(1)}(x_1, \ldots, x_n)\rangle \ldots |\tilde{f}^{(m)}(x_1, \ldots, x_n)\rangle\,, \tag{5.14}$$

which we recall will be in the state $|1^n\rangle$ if and only if the original m equations in n variables $f^{(1)}(x_1, \ldots, x_n), \ldots, f^{(m)}(x_1, \ldots, x_n)$ are satisfied by the assignment $x_1 \ldots x_n \in \{0,1\}^n$.
We can express any degree two equation $f^{(l)}(x_1, \ldots, x_n) \in \mathbb{F}_2[x_1, \ldots, x_n]$ where

$$f^{(l)}(x_1, \ldots, x_n) = c^{(l)} + \sum_{i=1}^{n} x_i y_i^{(l)} \qquad \text{and} \qquad y_i^{(l)} = b_i^{(l)} + \sum_{j=1}^{i-1} a_{j,i}^{(l)} x_j \tag{5.15}$$

as

$$f^{(l)}(x_1, \ldots, x_n) = f_1^{(l)}(x_1, \ldots, x_{n-k}) + f_2^{(l)}(x_1, \ldots, x_{n-k}, x_{n-k+1}, \ldots, x_n) \tag{5.16}$$

where

$$f_1^{(l)}(x_1, \ldots, x_{n-k}) = c^{(l)} + \sum_{i=1}^{n-k} x_i y_i^{(l)} \quad \text{and} \quad f_2^{(l)}(x_1, \ldots, x_{n-k}, x_{n-k+1}, \ldots, x_n) = \sum_{i=n-k+1}^{n} x_i y_i^{(l)}. \tag{5.17}$$

We therefore have that

$$g_2(x_1, \ldots, x_n) = f_2^{(1)}(x_1, \ldots, x_{n-k}, x_{n-k+1}, \ldots, x_n) \| \ldots \| f_2^{(m)}(x_1, \ldots, x_{n-k}, x_{n-k+1}, \ldots, x_n) \| 0, \tag{5.18}$$

where the last bit is the single bit used for computation of the $y_i^{(l)}$. If we apply the preprocessing step of PARENTS for a chosen $0 \leq k \leq n$, then it is clear that $g_1(x_1, \ldots, x_{n-k})$ remains unchanged, whilst for $z_{n-k+1} \ldots z_n \in \{0,1\}^k$ we have that

$$f_2^{(l)}(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n) = \sum_{i=n-k+1}^{n} z_i y_i^{(l)} = \sum_{i=n-k+1}^{n} z_i \cdot \left(b_i^{(l)} + \sum_{j=1}^{n-k} a_{j,i}^{(l)} x_j + \sum_{j=n-k+1}^{i-1} a_{j,i}^{(l)} z_j \right). \tag{5.19}$$

As $z_{n-k+1} \ldots z_n \in \{0,1\}^k$, we therefore have that $f_2^{(l)}(x_1, \ldots, x_{n-k}, z_{n-k+1}, \ldots, z_n)$ is simply a linear equation in $n - k$ variables, involving up to $n - k$ variables and 1 constant term.

We therefore have that $U_{\chi_{n-k}} \cdots U_{\chi_1}$ can be accomplished by m quantum bit oracles for m equations in $n-k$ variables, whilst the 2^k+1 unitaries $U_{g_2}^{(0)}, U_{g_2}^{(1)}, \ldots, U_{g_2}^{(2^k)}$ are each m additions of linear equations involving $n-k$ variables and 1 constant. U_{χ_*} remains as a $\wedge_m(X)$ gate with the m qubit equation register as the controls and the output of the quantum register as the target. Putting all this together we obtain the cost formula for Grover's algorithm in conjunction with the PARENTS approach by using the cost of the quantum bit oracle $\mathcal{O}_{\tilde{f}^{(l)}}$ from Table 2.4 and the other costs from Table 2.1 using the quantum bit oracles from for a total Clifford+T circuit size of

$$n + \left\lfloor \frac{\pi}{4} \cdot 2^{(n-k)/2} \right\rfloor \cdot \Big(2m((n-k)^2 + 18(n-k) - 20) + 40(n-k) - 60 + 4n \\ + (2^k+1)m(n-k+1) + 2^k(40m - 64) \Big). \tag{5.20}$$

5.1.3 Asymptotic and concrete results

As before, we can find the asymptotic complexity of this approach via simply taking the derivative of Equation (5.20) with respect to k. Once this derivative is set equal to 0 and solved for k, we obtain that the optimal value of k is on the order of $O\left(\log_2(n)\right)$. After substitution, we obtain that the asymptotic complexity of the PARENTS approach applied to instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ is $O\left(2^{n/2} m n^{3/2}\right)$. This does not improve upon the asymptotic complexity of the GENTS low-memory approach but does offer concrete advantages, as can be seen in Table 5.1. By numerical simulation and using the cost metric of total quantum circuit-size as an optimisation target, we obtain the following results by relying on our PARENTS to solve an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$.

λ	$\mathcal{MQ}(\mathbb{F}_2, n, m)$	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits	Success %	k	Oracle type
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{76.9}$	$2 \cdot 2^{73.46}$	$2 \cdot 2^{77.03}$	$2 \cdot 2^{73.23}$	$2 \cdot 2^{76.19}$	228	≈ 100.00	8	Low-memory [SW16]/PARENTS
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{78.35}$	$2 \cdot 2^{74.05}$	$2 \cdot 2^{78.43}$	$2 \cdot 2^{73.12}$	$2 \cdot 2^{77.23}$	236	≈ 100.00	5	Low-memory [SW16]/GENTS
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{79.89}$	$2 \cdot 2^{74.53}$	$2 \cdot 2^{79.97}$	$2 \cdot 2^{74.53}$	$2 \cdot 2^{79.89}$	236	≈ 100.00	N/A	Low-memory [SW16]
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{80.88}$	$2 \cdot 2^{76.76}$	$2 \cdot 2^{80.96}$	$2 \cdot 2^{75.57}$	$2 \cdot 2^{80.88}$	126	≈ 100.00	N/A	Counter-based [SW16]
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{124.96}$	$2 \cdot 2^{120.78}$	$2 \cdot 2^{125.04}$	$2 \cdot 2^{120.55}$	$2 \cdot 2^{124.04}$	411	≈ 100.00	9	Low-memory [SW16]/PARENTS
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{126.44}$	$2 \cdot 2^{121.32}$	$2 \cdot 2^{126.47}$	$2 \cdot 2^{120.41}$	$2 \cdot 2^{125.27}$	420	≈ 100.00	6	Low-memory [SW16]/GENTS
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{128.35}$	$2 \cdot 2^{123.38}$	$2 \cdot 2^{128.34}$	$2 \cdot 2^{122.18}$	$2 \cdot 2^{128.35}$	420	≈ 100.00	N/A	Low-memory [SW16]
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{129.34}$	$2 \cdot 2^{124.42}$	$2 \cdot 2^{129.39}$	$2 \cdot 2^{123.21}$	$2 \cdot 2^{129.34}$	219	≈ 100.00	N/A	Counter-based [SW16]
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{251.75}$	$2 \cdot 2^{246.45}$	$2 \cdot 2^{251.79}$	$2 \cdot 2^{246.20}$	$2 \cdot 2^{250.77}$	906	≈ 100.00	10	Low-memory [SW16]/PARENTS
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{253.25}$	$2 \cdot 2^{247.03}$	$2 \cdot 2^{253.26}$	$2 \cdot 2^{246.10}$	$2 \cdot 2^{252.26}$	916	≈ 100.00	7	Low-memory [SW16]/GENTS
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{255.69}$	$2 \cdot 2^{249.63}$	$2 \cdot 2^{255.72}$	$2 \cdot 2^{248.42}$	$2 \cdot 2^{255.69}$	916	≈ 100.00	N/A	Low-memory [SW16]
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{256.69}$	$2 \cdot 2^{250.65}$	$2 \cdot 2^{256.71}$	$2 \cdot 2^{249.44}$	$2 \cdot 2^{256.69}$	468	≈ 100.00	N/A	Counter-based [SW16]

Table 5.1: Quantum resource estimates for solving $\mathcal{MQ}(\mathbb{F}_2, n, m)$ using Grover and PARENTS.

As is becoming a theme, we break the proposed parameters for the Gui cryptosystem. There are several advantages to employing PARENTS, compared to GENTS. The first is that we strictly use fewer $\wedge_2(X)$ gates as we do not need to uncompute and recompute additions to the equation registers using these gates. This leads to a slightly smaller gate count and crucially, a lower T gate count. If we are optimising towards the T gate metric, then PARENTS has the edge over the ENTS and GENTS approaches in this metric. PARENTS also requires slightly fewer qubits.

5.2 Nested-quantum-search-and-memory

The concept of nesting of quantum search is well-known [CGW00], but this is usually exploited to make asymptotic gains with regards to the query complexity of algorithms — a quantum search algorithm that essentially works in an analogue of pruning a search tree. If the search problem is defined by the boolean indicator function is defined by $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and furthermore we have the decomposition

$$\chi(x_1 \dots x_n) \mapsto \chi_1(x_1 \dots x_k) \wedge \chi_2(x_1 \dots x_n) \tag{5.21}$$

where

$$\chi_1 : \{0,1\}^k \longrightarrow \{0,1\} \qquad \text{and} \qquad \chi_2 : \{0,1\}^n \longrightarrow \{0,1\} \tag{5.22}$$

then amplitude amplification (see Theorem 3.6) can be used to create a quantum algorithm that exploits this structure. We sketch this approach, which will be used later. We have used Grover's algorithm up til now and the reader unfamiliar with amplitude amplification is advised to revisit Definition 3.1 of the success probability of a boolean function relative to a quantum algorithm, our formulation of amplitude amplification in Theorem 3.6 as well as Definition 3.8 which gives the Hadamard transformation on n qubits and Theorem 3.7 (Quadratic Speedup).

5.3 A basic nested quantum search algorithm

Let the quantum phase oracles $\mathcal{O}_\chi$ and $\mathcal{O}_{\chi_1}$ be defined by the boolean indicator functions given above and assume that $M_\chi = |\chi^{-1}(1)| = 1$ and $M_{\chi_1} = |\chi_1^{-1}(1)|$, which is known. We define the initial quantum algorithm $\mathcal{A}_1 = H^{\otimes k} \otimes \mathcal{I}^{\otimes n-k}$, which has a success probability of $a \geq \frac{M_{\chi_1}}{2^k}$ relative to $\chi_1 : \{0,1\}^k \longrightarrow \{0,1\}$ if we consider only the first k qubits. By using amplitude amplification with Theorem 3.7, we can create a quantum algorithm $\mathcal{B}$ that succeeds with a success probability of $b \geq 1 - \frac{M_{\chi_1}}{2^k}$ relative to χ_1 and which requires $O(\sqrt{\frac{2^k}{M_{\chi_1}}})$ applications of the quantum phase oracle $\mathcal{O}_{\chi_1}$ and quantum algorithm $\mathcal{A}_1$. We will know b as we know M_{χ_1}.

We can then define $\mathcal{A}_2 = \mathcal{I}^{\otimes k} \otimes H^{\otimes n-k}$, and in turn define the new quantum algorithm $\mathcal{C}$, which involves first executing $\mathcal{B}$ to create a superposition of elements in the first k bit register which has a probability b of satisfying $\chi_1(x_1 \dots x_k) = 1$ and then using amplitude amplification on the second $n-k$ qubit register with the quantum algorithm $\mathcal{A}_2$ and quantum oracle $\mathcal{O}_\chi$. The quantum algorithm $\mathcal{C}$ therefore creates the state $|\varphi\rangle |\psi\rangle$, where $|\varphi\rangle$ has a probability of b of measuring $x_1 \dots x_k \in \{0,1\}^k$ such that $\chi_1(x_1 \dots x_k) = 1$. If we have measured such an element and then measure the second register $|\psi\rangle$ then we have a probability of $c' \geq 1 - \frac{1}{2^{n-k}}$ of obtaining bitstring $x_{k+1} \dots x_n \in \{0,1\}^{n-k}$ such that $\chi(x_1 \dots x_k x_{k+1} \dots x_n) = 1$. $\mathcal{C}$ costs 1 application of $\mathcal{B}$ and $\sqrt{2^{n-k}}$ applications of $\mathcal{A}_2$ and has a success probability of $\frac{bc'}{M_{\chi_1}} \approx \frac{1}{M_{\chi_1}}$ relative to χ.

We can therefore use amplitude amplification one final time in conjunction with $\mathcal{C}$ and $\mathcal{O}_\chi$ to create a new quantum algorithm $\mathcal{D}$ that costs $O\left(\sqrt{2^k} + \sqrt{M_{\chi_1}}\sqrt{2^{n-k}}\right)$ calls to the quantum oracles $\mathcal{O}_{\chi_1}$ and $\mathcal{O}_\chi$ and has an asymptotic success probability of 1.

If we compare this with Grover's algorithm, the best we could do would be to use the quantum phase oracle $\mathcal{O}_\chi$ for a cost of $O(\sqrt{2^n})$ or repeatedly search and sample using Grover's algorithm with the quantum phase oracle $\mathcal{O}_{\chi_1}$, then test the remaining $n-k$ variables using a Grover search with the quantum oracle $\mathcal{O}_\chi$. This sampling approach would therefore require on the order of $O(M_{\chi_1})$ such trials for an asymptotic cost of $O\left(\sqrt{M_{\chi_1}}\sqrt{2^k} + M_{\chi_1}\sqrt{2^{n-k}}\right)$ oracle queries — hence nesting quantum search procedures allows us to exploit information gained at levels of the search tree implicitly defined by the introduction of variables into computation.

5.3.1 Better memory management through nesting amplitude amplification

We use the nesting of quantum search procedures in a fundamentally different way, as no exploitable information concerning whether an assignment of any subset of variables is potentially part of a solution will be revealed for any subset of variables. Whilst our method bears similarity to that used by Arunachalamref and de Wolf [AdW17] in their technique to optimise Grover's algorithm in relation to the total number of quantum gates for other than those used in the quantum phase oracle — they treat the quantum phase oracle as a black-box. We require structure to make gains, using our tree decomposition of the quantum bit oracle (see Definition 4.11) and demonstrate that we can obtain computational gains above the ENTS, GENTS or PARENTS approaches in certain metrics via this method.

The GENTS and PARENTS approaches share a simple concept — we compute the quantum bit oracle up to a certain level using the sequence of unitary operators $U_{\chi_k} \cdots U_{\chi_1}$, then perform a secondary search from this level so that we can exploit the computational effort we have invested in executing $U_{\chi_k} \cdots U_{\chi_1}$. We can perform this same trick with amplitude amplification. The method can be treated more formally and given a completely recursive treatment, but unfortunately time pressures on the author did not permit including these proofs and results in the thesis. We present the method as a simple two-stage search procedure, which will capture the majority of the gains (through computational experiments, the number of levels never exceeded 3 and the gains were minor past 2 levels). We leave a full analysis of this for future work.

Theorem 5.1 (Making better use of memory via nested quantum search)**.**
Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $M_\chi = |\chi^{-1}(1)| = 1$ define a search problem and let $U_{\chi_1}, \ldots, U_{\chi_n}, U_{\chi_*}$ be a tree-decomposition (see Definition 4.11) of the quantum bit oracle $\mathcal{O}_\chi^{(b)}$, which uses w qubits for memory, n qubits for the search space and 1 qubit for the output.
Then there exists a quantum algorithm $\mathcal{B}_1$ which succeeds with probability

$$b_1 \geq 1 - \frac{b_2}{2^{n-k}} \geq 1 - \frac{1}{2^{n-k}} \tag{5.23}$$

relative to χ and which costs

$$E_{\mathcal{B}_1} = E_{H^{\otimes n-k}} + \sum_{i=1}^{n-k} E_{U_{\chi_i}} + E_{\mathcal{B}_2} + k_1 \cdot \left(2 \cdot \sum_{i=n-k+1}^{n} E_{U_{\chi_i}} + E_{U_{\chi_*}} + E_{\bar{n}} + 2 \cdot (E_{H^{\otimes n-k}} + \sum_{i=1}^{n-k} E_{U_{\chi_i}} + E_{\mathcal{B}_2}) \right) \tag{5.24}$$

where

$$E_{\mathcal{B}_2} = E_{H^{\otimes k}} + k_2 \left(E_{\chi_*} + 2 \cdot \sum_{i=n-k+1}^{n} E_{\chi_i} + E_{\bar{k}} + 2E_{H^{\otimes k}} \right) \tag{5.25}$$

and we have that

$$k_1 = \left\lfloor \frac{\pi}{4 \cdot \arcsin \sqrt{\frac{b_2}{2^{n-k}}}} \right\rfloor \qquad \text{and} \qquad k_2 = \left\lfloor \frac{\pi}{4 \cdot \arcsin \sqrt{\frac{1}{2^k}}} \right\rfloor, \tag{5.26}$$

and

$$b_2 = \sin^2 \left((2k_2 + 1) \cdot \arcsin \sqrt{\frac{1}{2^k}} \right). \tag{5.27}$$

Proof. We consider the application of a quantum algorithm

$$\mathcal{A}_1 = \mathcal{I}^w \otimes H^{\otimes n-k} \otimes \mathcal{I}^{k+1} \tag{5.28}$$

to the computational basis state (where $d \in \{0,1\}$)

$$|0^w\rangle \, |0^n\rangle \, |d\rangle \, . \tag{5.29}$$

This clearly results in the quantum state

$$\frac{1}{2^{(n-k)/2}} \sum_{x \in \{0,1\}^{n-k}} |0^w\rangle \, |x\rangle \, |0^k\rangle \, |d\rangle \, . \tag{5.30}$$

As $M_\chi = |\chi^{-1}(1)| = 1$, for any $0 < k < n$ there exists a single $x' \in \{0,1\}^{n-k}$ and $y' \in \{0,1\}^k$ such that $\chi(x'\|y') = 1$. If we examine the effect of a secondary algorithm

$$\mathcal{A}_2 = \mathcal{I}^{w+n-k} \otimes H^{\otimes k} \otimes \mathcal{I} \tag{5.31}$$

upon the computational basis state

$$|0^w\rangle \, |x'\rangle \, |0^k\rangle \, |d\rangle \, , \tag{5.32}$$

then this results in the state

$$\sum_{y \in \{0,1\}}^{k} |0^w\rangle \, |x'\rangle \, |y\rangle \, |d\rangle \, . \tag{5.33}$$

If we define the boolean function $\chi_{x'\|} : \{0,1\}^k \longrightarrow \{0,1\}$ by the action $\chi_{x'\|}(y) \mapsto \chi(x'\|y)$, then $\mathcal{A}_2$ has a success probability of $\frac{1}{2^k}$ relative to $\chi_{x'\|}$ for this computational basis state. We can therefore use amplitude amplification with $\mathcal{O}_{\chi_{x'\|}}$ and the quantum algorithm $\mathcal{A}_2$ to produce a quantum algorithm $\mathcal{B}_2$ that succeeds with success probability $b_2 \geq 1 - \frac{1}{2^k}$ relative to $\chi_{x'\|}$ on this computational basis state. As $\mathcal{O}_\chi = \mathcal{O}_{\chi_{x'\|}}$, we can write this quantum algorithm as

$$\mathcal{B}_2 = \left(\mathcal{A}_2 \mathcal{R}_k \mathcal{A}_2^\dagger \mathcal{O}_\chi \right)^{k_2} \mathcal{A}_2, \qquad \text{where} \qquad k_2 = \left\lfloor \frac{\pi}{4 \cdot \arcsin \sqrt{\frac{1}{2^k}}} \right\rfloor , \tag{5.34}$$

where $\mathcal{R}_k = \mathcal{I}^{\otimes w+n-k} \otimes -\mathcal{O}_{0^k} \otimes \mathcal{I}$. If we apply $\mathcal{B}_2 \mathcal{A}_1$ to the state (5.29), then we achieve a quantum algorithm with a success probability of $\frac{b_2}{2^{n-k}}$ relative to χ, as we have a $\frac{1}{2^{n-k}}$ chance of measuring the first $n-k$ qubits as x' and a chance of b_2 to measure y' if we have measured x'. We can then create the quantum algorithm $\mathcal{B}_1$ with success probability $b_1 \geq 1 - \frac{b_2}{2^{n-k}} \geq 1 - \frac{1}{2^{n-k}}$

$$\mathcal{B}_1 = \left(\mathcal{B}_2 \mathcal{A}_1 \mathcal{R}_n \mathcal{A}_1^\dagger \mathcal{B}_2^\dagger \mathcal{O}_\chi \right)^{k_1} \mathcal{B}_2 \mathcal{A}_1 \qquad \text{where} \qquad k_1 = \left\lfloor \frac{\pi}{4 \cdot \arcsin \sqrt{\frac{b_2}{2^{n-k}}}} \right\rfloor . \tag{5.35}$$

If we write this quantum algorithm out explicitly, then we get

$$\mathcal{B}_1 = \left(\left(\mathcal{A}_2\mathcal{R}_k\mathcal{A}_2^\dagger\mathcal{O}_\chi\right)^{k_2}\mathcal{A}_2\mathcal{A}_1\mathcal{R}_n\mathcal{A}_1^\dagger\mathcal{A}_2^\dagger\left(\mathcal{O}_\chi\mathcal{A}_2\mathcal{R}_k\mathcal{A}_2^\dagger\right)^{k_2}\mathcal{O}_\chi\right)^{k_1}\left(\mathcal{A}_2\mathcal{R}_k\mathcal{A}_2^\dagger\mathcal{O}_\chi\right)^{k_2}\mathcal{A}_2\mathcal{A}_1 \quad (5.36)$$

From here, we note that we can replace the quantum phase oracle with the tree-decomposition of quantum bit oracle. If we examine the effect of this upon the quantum algorithm $\mathcal{B}_2$, then we get that

$$\left(\mathcal{A}_2\mathcal{R}_k\mathcal{A}_2^\dagger\mathcal{O}_\chi\right)^{k_2}\mathcal{A}_2 = \left(\mathcal{A}_2\mathcal{R}_k\mathcal{A}_2^\dagger U_{\chi_1}^\dagger\cdots U_{\chi_{n-k}}^\dagger U_{\chi_{n-k+1}}^\dagger\cdots U_{\chi_n}^\dagger U_{\chi_*}U_{\chi_n}\cdots U_{\chi_{n-k+1}}U_{\chi_{n-k}}\cdots U_{\chi_1}\right)^{k_2}\mathcal{A}_2 \quad (5.37)$$

$$= U_{\chi_1}^\dagger\cdots U_{\chi_{n-k}}^\dagger\left(\mathcal{A}_2\mathcal{R}_k\mathcal{A}_2^\dagger U_{\chi_{n-k+1}}^\dagger\cdots U_{\chi_n}^\dagger U_{\chi_*}U_{\chi_n}\cdots U_{\chi_{n-k+1}}\right)^{k_2}\mathcal{A}_2 U_{\chi_{n-k}}\cdots U_{\chi_1} \quad (5.38)$$

as $\mathcal{A}_2$ introduces the last k variables, whereas $U_{\chi_{n-k}}\cdots U_{\chi_1}$ involves only the first $n-k$ variables. Because of this, we can rewrite $\mathcal{B}_1$ as

$$\mathcal{B}_1 = \left(\bar{\mathcal{B}}_2\bar{\mathcal{A}}_1\mathcal{R}_n\bar{\mathcal{A}}_1^\dagger\bar{\mathcal{B}}_2^\dagger U_{\chi_{n-k+1}}^\dagger\cdots U_{\chi_n}^\dagger U_{\chi_*}U_{\chi_n}\cdots U_{\chi_{n-k+1}}\right)^{k_1}\bar{\mathcal{B}}_2\bar{\mathcal{A}}_1 \qquad \text{where} \qquad \bar{\mathcal{A}}_1 = U_{\chi_{n-k}}\cdots U_{\chi_1}\mathcal{A}_1 \quad (5.39)$$

and

$$\bar{\mathcal{B}}_2 = \left(\mathcal{A}_2\mathcal{R}_k\mathcal{A}_2^\dagger U_{\chi_{n-k+1}}^\dagger\cdots U_{\chi_n}^\dagger U_{\chi_*}U_{\chi_n}\cdots U_{\chi_{n-k+1}}\right)^{k_2}\mathcal{A}_2 \quad (5.40)$$

In this way, we have pushed a portion of the cost to an earlier part of the quantum oracle computation which has a smaller query complexity — just as in the ENTS, GENTS and PARENTS approach. If we tally the costs, we therefore get that this approach has a cost of

$$E_{\mathcal{B}_1} = E_{H^{\otimes n-k}} + \sum_{i=1}^{n-k}E_{U_{\chi_i}} + E_{\bar{\mathcal{B}}_2} + k_1\cdot\left(2\cdot\sum_{i=n-k+1}^{n}E_{U_{\chi_i}} + E_{U_{\chi_*}} + E_{\bar{n}} + 2\cdot(E_{H^{\otimes n-k}} + \sum_{i=1}^{n-k}E_{U_{\chi_i}} + E_{\bar{\mathcal{B}}_2})\right) \quad (5.41)$$

where

$$E_{\bar{\mathcal{B}}_2} = E_{H^{\otimes k}} + k_2\left(E_{U_{\chi_*}} + 2\cdot\sum_{i=n-k+1}^{n}E_{U_{\chi_i}} + E_{\bar{k}} + 2E_{H^{\otimes k}}\right) \quad (5.42)$$

and we have that

$$k_1 = \left\lfloor\frac{\pi}{4\cdot\arcsin\sqrt{\frac{b_2}{2^{n-k}}}}\right\rfloor \qquad \text{and} \qquad k_2 = \left\lfloor\frac{\pi}{4\cdot\arcsin\sqrt{\frac{1}{2^k}}}\right\rfloor \quad (5.43)$$

where

$$b_2 = \sin^2\left((2k_2+1)\cdot\arcsin\sqrt{\frac{1}{2^k}}\right). \quad (5.44)$$

□

5.3.2 Asymptotic and concrete results

If we consider the higher-order terms for the low-memory approach and the high-memory approach as discussed with regards to the GENTS approach in Section 4.5 then we can derive their corresponding asymptotic complexities.

The complexity of the low-memory approach is on the order of $O\left(2^{(n-k)/2}mn^2 + 2^{n/2}mnk\right)$, hence taking the derivative with respect to k and solving the subsequent equation set equal to 0 for k gives us that the optimal value of k is on the order of $O\left(2\log_2 n\right)$. Substitution of this value of k gives us that the low-memory approach has an asymptotic complexity of $O\left(2^{n/2}mn\log_2 n\right)$.

The complexity of the high-memory approach is on the order of $O\left(2^{(n-k)/2}mn^2 + 2^{n/2}mk^2\right)$. We can use the same method as described above to derive that the optimal value of k is in the order of $O\left(4\log_2 n\right)$. Substitution into the original cost equation therefore gives us that the high-memory approach has an asymptotic complexity of $O\left(2^{n/2}m(\log_2 n)^2\right)$.

Table 5.2 below gives the concrete results with regards to the low-memory strategy and the high memory strategy, as discussed above.

λ	$\mathcal{MQ}(\mathbb{F}_2, n, m)$	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits	Success %	k	Oracle type
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{77.86}$	$2 \cdot 2^{76.21}$	$2 \cdot 2^{78.26}$	$2 \cdot 2^{74.99}$	$2 \cdot 2^{77.86}$	236	≈ 100.00	7	Low-memory [SW16]/Theorem 5.1
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{75.03}$	$2 \cdot 2^{73.39}$	$2 \cdot 2^{75.43}$	$2 \cdot 2^{72.32}$	$2 \cdot 2^{72.39}$	1756	≈ 100.00	13	High-memory [SW16]/Theorem 5.1
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{79.89}$	$2 \cdot 2^{74.53}$	$2 \cdot 2^{79.97}$	$2 \cdot 2^{74.53}$	$2 \cdot 2^{79.89}$	236	≈ 100.00	N/A	Low-memory [SW16]
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{80.88}$	$2 \cdot 2^{76.76}$	$2 \cdot 2^{80.96}$	$2 \cdot 2^{75.57}$	$2 \cdot 2^{80.88}$	126	≈ 100.00	N/A	Counter-based [SW16]
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{125.76}$	$2 \cdot 2^{123.9}$	$2 \cdot 2^{126.12}$	$2 \cdot 2^{123.95}$	$2 \cdot 2^{125.77}$	420	≈ 100.00	9	Low-memory [SW16]/Theorem 5.1
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{122.19}$	$2 \cdot 2^{120.46}$	$2 \cdot 2^{122.58}$	$2 \cdot 2^{119.37}$	$2 \cdot 2^{119.15}$	3763	≈ 100.00	16	High-memory [SW16]/Theorem 5.1
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{128.35}$	$2 \cdot 2^{123.38}$	$2 \cdot 2^{128.34}$	$2 \cdot 2^{122.18}$	$2 \cdot 2^{128.35}$	420	≈ 100.00	N/A	Low-memory [SW16]
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{129.34}$	$2 \cdot 2^{124.42}$	$2 \cdot 2^{129.39}$	$2 \cdot 2^{123.21}$	$2 \cdot 2^{129.34}$	219	≈ 100.00	N/A	Counter-based [SW16]
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{252.29}$	$2 \cdot 2^{250.29}$	$2 \cdot 2^{252.60}$	$2 \cdot 2^{249.20}$	$2 \cdot 2^{252.29}$	916	≈ 100.00	11	Low-memory [SW16]/Theorem 5.1
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{247.73}$	$2 \cdot 2^{244.75}$	$2 \cdot 2^{248.08}$	$2 \cdot 2^{244.75}$	$2 \cdot 2^{244.18}$	10055	≈ 100.00	20	High-memory [SW16]/Theorem 5.1
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{255.69}$	$2 \cdot 2^{249.63}$	$2 \cdot 2^{255.72}$	$2 \cdot 2^{248.42}$	$2 \cdot 2^{255.69}$	916	≈ 100.00	N/A	Low-memory [SW16]
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{256.69}$	$2 \cdot 2^{250.65}$	$2 \cdot 2^{256.71}$	$2 \cdot 2^{249.44}$	$2 \cdot 2^{256.69}$	468	≈ 100.00	N/A	Counter-based [SW16]

Table 5.2: Quantum resource estimates for solving $\mathcal{MQ}(\mathbb{F}_2, n, m)$ using Theorem 5.1.

Chapter 6

Improvements to the Search with Two Oracles approach

If a man will begin with certainties, he shall end in doubts; but if he will be content to begin with doubts he shall end in certainties.

- Francis Bacon, *The Advancement of Learning [Bac05]*

In this chapter we examine how the quantum search algorithm put forth by Kimmel et al. [KYYLHH15], designed to lower the quantum resources required when we have multiple quantum search oracles at our disposal, can be adapted to real-world problems, such as quantum cryptanalysis of the Advanced Encryption Standard and the Multivariate Quadratic hardness assumption. Their technique, posed as a solution to a problem entitled *Search with Two Oracles* (*STO*), is designed to reduce the total cost of quantum search when there exists one quantum search oracle which is expensive but marks the single target we are searching for, whilst the other quantum search oracle is relatively cheap and marks a subset of the search-space which includes this target, and we know the size of this larger subset precisely.

The main contribution of this chapter is demonstrating how their technique can be adapted to lower quantum resource estimates for quantum search applied to these problems when there is uncertainty in the number of additional targets that the cheaper oracle marks. In particular we note that in this scenario we can obtain the cost benefits of their approach, without increasing the number of qubits required, by artifically ensuring that the cheaper oracle marks a large number of bitstrings which we know are not solutions to the problem instance. By introducing these false targets into the search space we ensure that the ratio between the true number of targets marked by this oracle and the guessed number of targets marked by this oracle is close to 1 as the number of false targets increases. In concrete terms, this allows us to lower the quantum resource estimates for applying quantum search to instances of the key-search problem for the Advanced Encryption Standard (AES) and in solving instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$, particularly with regards to low-qubit implementations of quantum search for these problems.

These gains are modest, but significant, in that they outperform Grover's algorithm naively applied to these problems [GLRS16, SW16] and the quantum resources required to perform cryptanalysis of AES are relevant to the ongoing NIST standardisation process for quantum-resistant public-key cryptosystems [oST16b, oST16a]. That the low-qubit $\mathcal{MQ}$ oracle requires only $n + \lceil \log_2 m \rceil + 1$ qubits to implement means that this oracle may be one of the first applications of quantum search that is possible to implement, even if asymptotically more efficient designs are possible when greater number of qubits are available [BY18, FHK$^+$17].

In Section 6.1 we discuss the *Search with Two Oracles* (*STO*) problem [KYYLHH15] and the complications in applying their method to several problems in cryptanalysis, that of applying *STO* to real-world problems. In Section 6.2 we discuss how, given these problems, we can adapt their method and in Section 6.3 we conclude with new resource estimates for the single-target quantum cryptanalysis of the Advanced Encryption Standard (AES) and low-qubit quantum cryptanalysis of the Multivariate Quadratic problem over $\mathbb{F}_2$.

6.1 Reviewing Oracles with Costs

We now review the *Search with Two Oracles* (*STO*) problem and its cost-effective solution as proposed by Kimmel et al. [KYYLHH15]. We make a minor modification to their results, in that we include additional costs other than the quantum oracle. These changes preserve their results and capture the use-case of our extension to the solution to the *STO* problem. Our results will follow naturally from this formulation. In all cases, cost may be taken to mean either circuit-depth or circuit-size.

6.1.1 The *STO* problem and a solution

Definition 6.1 (Search with Two Oracles (*STO*) [KYYLHH15])**.**
Let $f_*, f_S : \{0,1\}^n \longrightarrow \{0,1\}$ be two boolean functions with the property that

$$f_*^{-1}(1) \subseteq f_S^{-1}(1) \quad \text{where} \quad M_* = |f_*^{-1}(1)| \in \{0,1\} \ \text{ and } \ M_S = |f_S^{-1}(1)| \tag{6.1}$$

and which respectively define the quantum bit oracles $\mathcal{O}_{f_*}^{(b)}$ and $\mathcal{O}_{f_S}^{(b)}$.
The *Search with Two Oracles (STO)* problem is to locate an element $x \in \{0,1\}^n$ such that $f_*(x) = 1$ or prove that no such element exists. It is given that $E_{\mathcal{O}_*} \geq E_{\mathcal{O}_S}$.

Kimmel et al. [KYYLHH15] provide us with the promise that we know the exact value of M_S. This is not a realistic scenario for most problems and so in this section we examine both their solution and its behaviour when we guess that $M_S = M_S'$.

6.1.2 A classical solution to the *STO* problem

A classical solution to the *STO problem* is to simply perform an exhaustive search on the domain $\{0,1\}^n$ with f_S and when we have obtained an element such that $f_S(x) = 1$, we test it to check whether $f_*(x) = 1$. This is akin to a filtering process and both the *STO* solution suggested by Kimmel et al. and our own adaptation use this basic idea — though our adaptation is somewhat harder to draw a classical analogy for.

The complexity of the above approach can be computed — we can expect to sample $\frac{2^n+1}{2}$ elements until we have located the single element $x \in \{0,1\}^n$ that satisfies $f_*(x) = 1$, just as in the case of naive exhaustive search. Of these elements, we can expect that $\frac{M_S}{2^n}$ satisfy $f_S(x) = 1$. The total expected cost of the above approach will therefore be

$$\frac{2^n+1}{2} \cdot \left(\mathrm{Cost}(f_S) + \frac{M_S}{2^n} \cdot \mathrm{Cost}(f_*) \right) \tag{6.2}$$

whilst the expected cost of using a naive exhaustive search with f_* would be $\frac{2^n+1}{2} \cdot \mathrm{Cost}(f_*)$. This approach forms the basis of the solution of Kimmel et al. to the *STO* problem. We note that if $\mathrm{Cost}(f_S) = \mathrm{Cost}(f_*)$ then the above approach offers no advantage.

6.1.3 Previous quantum solutions to the *STO* problem

Kimmel et al. offer three algorithms to solve *STO*. If $E_* = E_S$, then we simply use Grover's algorithm with the quantum oracle $\mathcal{O}_*$ and ignore the quantum oracle $\mathcal{O}_S$, whilst if $E_* > E_S$ we use the quantum algorithm we describe below. The third algorithm interpolates between these two quantum algorithms, but is not relevant to the adaptation we present in this chapter.

We now describe their solution and give its approximate costs, then examine its correctness under potential real-world use cases where we can only guess that $M_S = M'_S$. It will be easily seen that the cost of our modification is both strictly less than the solution of Kimmel et al. and is easier to describe in closed form, as it can precisely be described in terms of the Clifford+T gate set without having to rely upon the approximation of the arbitrary single-qubit unitary transformations that exact amplitude amplification requires.

Theorem 6.2 (Solution to the *STO* problem using exact amplitude amplification [KYYLHH15])**.** Let $f_s, f_* : \{0,1\}^n \longrightarrow \{0,1\}$ and M_*, M_S define an instance of the *STO* problem.
There exists a quantum algorithm with a success probability of 1 relative to f_* and which has an asymptotic cost in terms of the oracles $\mathcal{O}_{f_S}^{(b)}$ and $\mathcal{O}_{f_*}^{(b)}$ of $O(2^{n/2}E_{f_S} + \sqrt{M_S}E_{f_*})$.

Proof. In the following, we allow $\mathcal{Z}_1$ and $\mathcal{Z}_2$ to be arbitrary single-qubit unitary transformations used to enable exact amplitude amplification. The functions $\hat{f}_S$ and $\hat{f}_*$ are defined as in the proof of Theorem 3.11 and we make the assumption that the quantum bit oracles $\mathcal{O}_{\hat{f}_S}^{(b)}$ and $\mathcal{O}_{\hat{f}_*}^{(b)}$ cost approximately the same as the quantum bit oracles $\mathcal{O}_{f_S}^{(b)}$ and $\mathcal{O}_{f_*}^{(b)}$. This is easily seen to be true in the case of the single-target preimage search problem defined by $h : \{0,1\}^n \longrightarrow \{0,1\}^m$ and $y_h \in \{0,1\}^m$, as if $|h(x) \stackrel{?}{=} y_h\rangle$ is written via a $\wedge_m(X)$ gate then we simply use a $\wedge_{m+1}(X)$ gate instead to write $|\big(h(x) \stackrel{?}{=} y_h\big) \wedge \big(x_{n+1} \stackrel{?}{=} 1\big)\rangle$.
We first define the quantum algorithm $\mathcal{A} = H^{\otimes n}$ (the Walsh-Hadamard transform) and use exact amplitude amplification (see Theorem 3.11) to create a quantum algorithm $\mathcal{B}$ with a success probability of 1 relative to the function f_S. By Theorem 3.11, $\mathcal{B}$ has a cost of

$$E_{\mathcal{B}} = E_{H^{\otimes n}\otimes\mathcal{Z}_1} + \left\lceil \frac{\pi}{4\arcsin\sqrt{\frac{M_S}{2^n}}} \right\rceil \cdot \left(E_{\hat{f}_S} + E_{\overline{n+1}} + 2E_{H^{\otimes n}\otimes\mathcal{Z}_1}\right) \tag{6.3}$$

We can then define a second quantum algorithm, $\mathcal{C}$, by using exact amplitude amplification with the quantum algorithm set to be $\mathcal{B}$, which has a success probability relative to f_* of $\frac{1}{M_S}$. By Theorem 3.6, we can create a quantum algorithm $\mathcal{C}$ with a success probability of 1 relative to f_*. By Theorem 3.11, $\mathcal{C}$ has a cost of

$$E_{\mathcal{C}} = E_{\mathcal{B}\otimes\mathcal{Z}_2} + \left\lceil \frac{\pi}{4\arcsin\sqrt{\frac{1}{M_S}}} \right\rceil \cdot \left(E_{\hat{f}_*} + E_{\overline{n+1}} + 2E_{\mathcal{B}\otimes\mathcal{Z}_2}\right). \tag{6.4}$$

Using the fact that $\arcsin x \approx x$ and that $E_{\hat{f}} \approx E_f$ therefore gives us that the approximate cost of this algorithm in terms of queries to the quantum bit oracles $\mathcal{O}_{f_S}^{(b)}$ and $\mathcal{O}_{f_*}^{(b)}$ is

$$\frac{\pi}{4}\sqrt{M_S} \cdot E_{f_*} + \frac{\pi^2}{8}\sqrt{2^n} \cdot E_{f_S}. \tag{6.5}$$

□

In essence, the query complexity remains identical to Grover as we are making $\mathcal{O}\big(\sqrt{2^n}\big)$ queries, but relative to a less expensive oracle. However, this asymptotic notation hides the constant in front of E_{f_S}, which is an important real-world factor and hinders (but does not rule out) a recursive strategy from being efficient if we possess $f_{S_i} : \{0,1\}^n \longrightarrow \{0,1\}$ such that $f_*^{-1}(1) \subseteq f_{S_1}^{-1}(1)$ and $f_{S_i}^{-1}(1) \subseteq f_{S_{i+1}}^{-1}(1)$. We will later provide an exact cost for a modified version of this algorithm exploiting only exact amplitude amplification in Section 6.2. Now that the costs have been discussed, we turn to the correctness of this algorithm under real-world conditions, where we do not precisely know the value of M_S but can only guess that $M_S = M_S'$.

6.1.4 On the probability of success of the solution to the STO problem

Theorem 3.11 describes the behaviour of one implementation of exact amplitude amplification when we make the wrong guess for the success probability of the quantum algorithm $\mathcal{A}$ relative to the boolean function χ. We can therefore use this to obtain an expression which gives us the success probability of the solution to the STO problem described in Theorem 6.2 when we make make the guess that $M_S = M'_S$. We will retain the promise that $M_* = 1$.

Theorem 6.3 (Success probability of the STO solution under the assumption $M_S = M'_s$)**.** Let $f_s, f_* : \{0,1\}^n \longrightarrow \{0,1\}$ and M_*, M_S define an instance of the STO problem. Suppose we guess $M_S = M'_S$. The algorithm in Theorem 6.2 has a success probability of

$$c = \sin^2\left(\left(2\hat{k}_2 + 1\right) \cdot \arcsin\sqrt{z \cdot \frac{M'_S}{M_S} \cdot \sin^2\left(\frac{\pi}{4\hat{k}_2 + 2}\right)}\right) \cdot \left(\frac{b_g - b \cdot b_g}{b_g - b \cdot \hat{b}_g}\right) + \frac{b \cdot b_g - b \cdot \hat{b}_g}{b_g - b \cdot \hat{b}_g} \quad (6.6)$$

where $b_g = \frac{1}{M'_S}$, $\hat{k}_2 = \left\lceil \frac{\pi}{4 \arcsin\sqrt{b_g}} \right\rceil$, $\hat{b}_g = \sin^2\left(\frac{\pi}{4\hat{k}_2+2}\right)$, $b = \frac{z}{M_S}$ and where

$$z = \sin^2\left(\left(2\hat{k}_1 + 1\right) \cdot \arcsin\sqrt{\frac{M_S}{M'_S} \cdot \sin^2\left(\frac{\pi}{4\hat{k}_1 + 2}\right)}\right) \cdot \left(\frac{a_g - a \cdot a_g}{a_g - a \cdot \hat{a}_g}\right) + \frac{a \cdot a_g - a \cdot \hat{a}_g}{a_g - a \cdot \hat{a}_g} \quad (6.7)$$

where $a_g = \frac{M'_S}{2^n}$, $\hat{k}_1 = \left\lceil \frac{\pi}{4 \arcsin\sqrt{a_g}} \right\rceil$, $\hat{a}_g = \sin^2\left(\frac{\pi}{4\hat{k}_1+2}\right)$ and $a = \frac{M_S}{2^n}$.

Proof. Under the assumption that $M_S = M'_S$, then we have guessed that the quantum algorithm $\mathcal{A} = H^{\otimes n}$ has a success probability of $a_g = \frac{M'_S}{2^n}$, when in reality it is $a = \frac{M_S}{2^n}$. By Theorem 3.11, the algorithm $\mathcal{B}$ as described in the proof of Theorem 6.2 has a success probability of z relative to the boolean function f_S and a success probability of $b = \frac{z}{M_S}$ relative to f_*. Our guess for the success probability of $\mathcal{B}$ relative to f_S is 1 and our guess for the success probability of $\mathcal{B}$ relative to f_* is therefore $b_g = \frac{1}{M'_S}$.

By using Theorem 3.11 again, we have that algorithm $\mathcal{C}$ as described in Theorem 6.2 has a probability of success of c relative to f_*. □

We now have a computational method of examining the projected success probability of algorithm $\mathcal{C}$ under the potentially erroneous assumption that $M_S = M'_S$. We note that if we have guessed correctly, then it is easily seen by substitution that $z = 1$ and $c = 1$ as in Theorem 6.2. We demonstrate the how the success probability of $\mathcal{C}$ changes relative to various values of M_S and M'_S in Figure 6-1 on the next page.

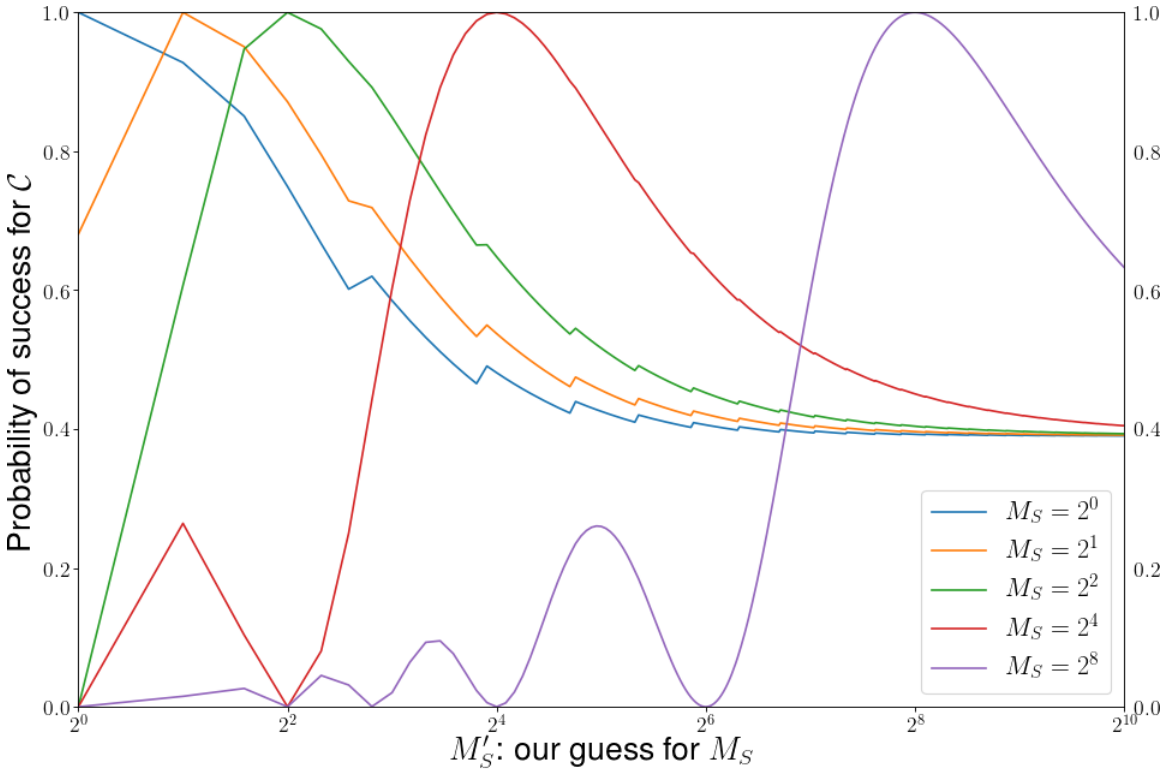


Figure 6-1: Success of $\mathcal{C}$ with $n = 128$, $M_* = 1$ and varied M_S.

As can be seen from Figure 6-1, the solution of Kimmel et al. can fail if we have guessed wrongly that $M_S = M'_S$. The success probability is not reliant upon the absolute value of the difference $|M_S - M'_S|$, but is instead reliant upon the *ratio* $M_S : M'_S$. This is easily observed by examining equations (6.6) and (6.7) above, as the quantity $\frac{M_S}{M'_S}$ affects the probability z and both the quantities z and $\frac{M'_S}{M_S}$ impact upon c, the success probability of $\mathcal{C}$ relative to f_*.

Quantum counting [BHT98, BHMT02] in conjunction with the oracle $\mathcal{O}_{f_S}$ can determine M_S to the required precision. Whilst obtaining the correct value of M_S obviously ensures that algorithm $\mathcal{C}$ succeeds with probability 1 relative to f_*, this approach may wipe out the computational gains and the quantum counting must be performed before we execute $\mathcal{C}$. We could alternatively simply run the computation multiple times, but this again increases the cost. We suggest a modification in Section 6.2 which recovers the correctness and computational gains.

In the modification we present in Section 6.2, we exploit the fact that it is the ratio $M_S : M'_S$ that affects the success probability of $\mathcal{C}$ relative to f_* and artificially modify this ratio to restore the correctness of this approach to solving the *STO* problem whilst retaining the computational advantages offered by this algorithm.

6.2 A modified solution to the STO problem

We now turn to our modified STO algorithm. The algorithm is essentially the solution of Kimmel et al. with two modifications that take into account the fact that we are only guessing at M_S and may only know the discrete probability distribution $\Pr[M_S = X]$ rather than the exact value of M_S. The first modification is minor — as the point of using exact amplitude amplification is to achieve a deterministic quantum algororithm with a success probability of 1 relative to f_* and the probability that we have guessed the correct value of M_S may be small, we will simply use amplitude amplification (see Theorem 3.6) instead of *exact* amplitude amplification (see Theorem 3.11). This simplifies the analysis and the modified algorithm can be implemented exactly by any universal quantum gate set that can implement Grover's algorithm exactly.

The second modification is the core result and allows us to increase the success probability of the algorithm by changing the ratio $M_S : M'_S$. This is done by noting that fact that the ratio $2^t + M_s : 2^t + M'_S$ approaches 1 when $2^t > M_S$ and as $t \to 2^n$. Thus, if we can instead modify the cheaper quantum oracle to work with this ratio instead of $M_S : M'_S$, then we can tame the success probability when only the discrete probability distribution $\Pr[M_S = X]$ is known.

Theorem 6.4 (A modified solution to the STO problem)**.**
Let $f_S, f_* : \{0,1\}^n \longrightarrow \{0,1\}$, $M_* \in \{0,1\}$ and $M_S = |f_S^{-1}(1)|$ define an instance of the STO problem, where M_S is unknown. Let $M'_S \in \mathbb{N}$ and $\epsilon \in [0,1]$ be such that $\Pr[M_S \geq M'_S] \leq \epsilon$ and let $t \in \mathbb{N}$ such that $0 \leq t \leq n$. Then there exists a quantum algorithm that terminates with a success probability relative to f_* greater than

$$(1-\epsilon) \cdot \sin^2\left(\left(\frac{\pi}{2\arcsin\sqrt{\frac{1}{2^t}}} - 2\right) \cdot \sqrt{\frac{b'}{M'_S + 2^t}}\right) \tag{6.8}$$

where

$$b' = \sin^2\left(\left(\frac{\pi}{2\arcsin\sqrt{\frac{M'_S+2^t}{2^n}}} - 2\right) \cdot \sqrt{\frac{2^t}{2^n}}\right). \tag{6.9}$$

The algorithm has a total execution cost of

$$\begin{aligned} & k_2\left(E_{\mathcal{O}_{f_*}} + E_{\bar{n}}\right) \\ & + (2_{k_1}+1)(2k_2+1)E_{H^{\otimes n}} \\ & + (2k_2+1)k_1(2E_{\mathcal{E}_{f_S}} + E_{\wedge_{n-t}(X)} + E_{\wedge_{n-t+1}(X)} + E_{\wedge_1(X)} + E_{\bar{n}}) \end{aligned} \tag{6.10}$$

where

$$k_1 = \left\lfloor \frac{\pi}{4\arcsin\sqrt{\frac{M'_S+2^t}{2^n}}} - \frac{1}{2} \right\rfloor \qquad \text{and} \qquad k_2 = \left\lfloor \frac{\pi}{4\arcsin\sqrt{\frac{1}{2^t}}} - \frac{1}{2} \right\rfloor. \tag{6.11}$$

Proof. Let $S = f_S^{-1(1)}$ and $Z_t = \left\{1^{n-t} \| x' : x' \in \{0,1\}^t\right\}$ be the set of 2^t bitstrings of length n whose first $n-t$ values are 1. We now define the function $f_{S \cup Z_t} : \{0,1\}^n \longrightarrow \{0,1\}$ by

$$f_{S \cup Z_t}(x) \mapsto \begin{cases} 1 & (f_S(x) = 1) \vee (x \in Z_t) \\ 0 & \text{otherwise} \end{cases} \tag{6.12}$$

and use the notation $M_{S \cup Z_t} = |f_{S \cup Z_t}^{-1}(1)|$. This naturally defines the quantum phase oracle $\mathcal{O}_{f_{S \cup Z_t}}$, which will replace the role of the quantum phase oracle $\mathcal{O}_{f_S}$. We will detail how $\mathcal{O}_{f_{S \cup Z_t}}$ may be constructed from the quantum evaluation $\mathcal{E}_{f_S}$ and prove the computational resources required for our modification after first proving a computational lower bound on the success probability of our modification.

Our first step is to define the quantum algorithm $\mathcal{B}$ by using amplitude amplification (see Theorem 3.6) with the quantum algorithm $\mathcal{A} = H^{\otimes n}$, the quantum phase oracle $\mathcal{O}_{f_{S \cup Z_t}}$ and setting $k_1 = \left\lfloor \frac{\pi}{4 \arcsin \sqrt{\frac{M'_S + 2^t}{2^n}}} - \frac{1}{2} \right\rfloor$ to create a quantum algorithm $\mathcal{B}$ with success probability

$$b = \sin^2\left(\left(2\left\lfloor \frac{\pi}{4 \arcsin \sqrt{\frac{M'_S + 2^t}{2^n}}} - \frac{1}{2} \right\rfloor + 1\right) \cdot \arcsin \sqrt{\frac{M_{S \cup Z_t}}{2^n}}\right) \tag{6.13}$$

$$> \sin^2\left(\left(\frac{\pi}{2 \arcsin \sqrt{\frac{M'_S + 2^t}{2^n}}} - 2\right) \cdot \sqrt{\frac{2^t}{2^n}}\right) = b' \tag{6.14}$$

by using the fact that $x \leq \arcsin x$, $x - 1 < \lfloor x \rfloor$ and that the argument of sine in (6.13) is less than $\frac{\pi}{2}$, hence we can exploit the fact that sine is an increasing function on the domain $[0, \frac{\pi}{2}]$.

We therefore have a computational lower bound b' on the success probability of $\mathcal{B}$ relative to $f_{S \cup Z_t}$. The success probability of $\mathcal{B}$ relative to f_* is therefore $\frac{b}{M_{S \cup Z_t}}$ and our lower bound on the success probability of $\mathcal{B}$ relative to f_* is $\frac{b'}{M'_S + 2^t}$.

Amplitude amplification with the quantum algorithm $\mathcal{B}$, quantum phase oracle $\mathcal{O}_{f_*}$ and with $k_2 = \left\lfloor \frac{\pi}{4 \arcsin \sqrt{\frac{1}{2^t}}} - \frac{1}{2} \right\rfloor$ then gives us a quantum algorithm $\mathcal{C}$ with a success probability

$$c = \sin^2\left(\left(2\left\lfloor \frac{\pi}{4 \arcsin \sqrt{\frac{1}{2^t}}} - \frac{1}{2} \right\rfloor + 1\right) \cdot \arcsin \sqrt{\frac{b}{M_{S \cup Z_t}}}\right) \tag{6.15}$$

$$> \sin^2\left(\left(\frac{\pi}{2 \arcsin \sqrt{\frac{1}{2^t}}} - 2\right) \cdot \sqrt{\frac{b'}{M'_S + 2^t}}\right) = c' \tag{6.16}$$

relative to f_* using the same strategy and arguments used to derive that $b > b'$ in conjunction with the fact that $\frac{b'}{M'_S + 2^t} \leq \frac{b}{M_{S \cup Z_t}} \leq \frac{1}{2^t}$. We therefore have a computational lower-bound on the success probability of $\mathcal{C}$ assuming that $M_S < M'_S$. The probability of this occurring is at least $1 - \epsilon$ by assumption, hence the success probability of $\mathcal{C}$ relative to f_* is greater than $(1 - \epsilon) \cdot c'$.

Implementing $\mathcal{O}_{f_{S\cup Z_t}}$. We assume that we possess a circuit to compute the quantum evaluation $\mathcal{E}_{f_S}$, though the following can be adapted to create the quantum bit oracle $\mathcal{O}^{(b)}_{f_{S\cup Z_t}}$ from the quantum bit oracle $\mathcal{O}^{(b)}_{f_S}$. The identity $A \vee B \equiv A \oplus B \oplus (A \wedge B)$ implies we can implement $f_{S\cup Z_t} : \{0,1\}^n \longrightarrow \{0,1\}$ via computing

$$f_{S\cup Z_t}(x) = f_S(x) \oplus (x \in Z_t) \oplus (f_S(x) \wedge (x \in Z_t)) \,. \tag{6.17}$$

We can compute $\mathcal{O}^{(b)}_{f_{S\cup Z_t}}$ with a single ancilla qubit. We first compute and store $|f_S(x)\rangle$ in this ancilla qubit via one quantum evaluation $\mathcal{E}_{f_S}$ and write it to the output qubit using a single $\wedge_1(X)$ gate. We can then write $|(x \stackrel{?}{\in} Z_t)\rangle$ to the output qubit via a single $\wedge_{n-t}(X)$ gate with the controls set to be the first $n-t$ qubits of $|x\rangle$. The addition of $f_S(X) \wedge (x \in Z_t)$ is performed via a single $\wedge_{n-t+1}(X)$ gate with the controls set to be the first $n-t$ qubits of $|x\rangle$ as before and the additional control being the qubit holding $|f_S(x)\rangle$. The output register then holds

$$|x\rangle\,|g(x)\rangle\,|f_S(x)\rangle\,|f_S(x) \oplus (x \in Z_t) \oplus (f_S(x) \wedge (x \in Z_t))\rangle \tag{6.18}$$

and we need only execute $\mathcal{E}^\dagger_{f_S}$ (the quantum evaluation $\mathcal{E}_{f_S}$ in reverse) to clear up the working memory used to compute $|f_S(x)\rangle$. Assuming that we have a single qubit kept in the state $|-\rangle$, the quantum phase oracle $\mathcal{O}_{f_{S\cup Z_t}}$ therefore has an execution cost of

$$E_{\mathcal{O}_{f_{S\cup Z_t}}} = 2E_{\mathcal{E}_{f_S}} + E_{\wedge_{n-t}(X)} + E_{\wedge_{n-t+1}(X)} + E_{\wedge_1(X)} \tag{6.19}$$

and as a $\wedge_k(X)$ gate can be implemented with $O(k)$ gates, it is easily seen that $E_{\mathcal{O}_{f_{S\cup Z_t}}} \approx E_{\mathcal{O}_{f_S}}$ if we are working with quantum evaluations as we can implement a quantum bit oracle via two quantum evaluations and one $\wedge_1(X)$ gate.

The cost of $\mathcal{B}$. Letting $k_1 = \left\lfloor \frac{\pi}{4 \arcsin \sqrt{\frac{M'_S + 2^t}{2^n}}} - \frac{1}{2} \right\rfloor$, we have that

$$E_{\mathcal{B}} = E_{H^{\otimes n}} + k_1 \cdot \left(2E_{\mathcal{E}_{f_S}} + \wedge_1(X) + E_{\wedge_{n-t}(X)} + E_{\wedge_{n-t+1}(X)} + E_{\bar{n}} + 2E_{H^{\otimes n}} \right). \tag{6.20}$$

The cost of $\mathcal{C}$. Letting $k_2 = \left\lfloor \frac{\pi}{4 \arcsin \sqrt{\frac{1}{2^t}}} - \frac{1}{2} \right\rfloor$, we have that

$$\begin{aligned}
E_{\mathcal{C}} &= E_{\mathcal{B}} + k_2 \left(E_{\mathcal{O}_{f_*}} + E_{\bar{n}} + 2E_{\mathcal{B}} \right) && (6.21)\\
&= k_2 \left(E_{\mathcal{O}_{f_*}} + E_{\bar{n}} \right) && (6.22)\\
&\quad + (2_{k_1} + 1)(2k_2 + 1) E_{H^{\otimes n}} && (6.23)\\
&\quad + (2k_2 + 1) k_1 \left(2E_{\mathcal{E}_{f_S}} + E_{\wedge_{n-t}(X)} + E_{\wedge_{n-t+1}(X)} + E_{\wedge_1(X)} + E_{\bar{n}} \right). && (6.24)
\end{aligned}$$

Putting these execution costs together, we have the total cost as given in Theorem 6.4. □

Using a first order approximation $\arcsin x \approx x$, if $M'_S \ll 2^t \ll 2^n$, it is evident that

$$k_1 \approx \frac{\pi}{4} \cdot 2^{(n-t)/2}, \qquad k_2 \approx \frac{\pi}{4} \cdot 2^{t/2} \qquad \text{and} \qquad (2k_2+1)k_1 \approx \frac{\pi^2}{8} \cdot 2^{n/2}. \tag{6.25}$$

Under the assumption that $2E_{\mathcal{E}_{f_{S\cup Z_t}}} \approx E_{\mathcal{O}_{f_{S\cup Z_t}}}$ (as it is for the single-target preimage search problem) and that $E_{H^{\otimes n}} \ll E_{\mathcal{O}_{f_S}} \ll E_{\mathcal{O}_{f_*}}$, we have that the term $\frac{\pi^2}{8} \cdot 2^{n/2} \cdot E_{\mathcal{O}_{f_S}}$ will dominate the execution cost, as in the original solution to the *STO* problem given in Theorem 6.3.

6.2.1 Advantages and disadvantages of the *STO* approach

Our main concern is the circuit-size of the quantum circuit, but circuit-depth is also an important metric. We briefly discuss the impact of our approach upon quantum circuit-depth. If $E_{\mathcal{O}_{f_*}}$ and $E_{\mathcal{O}_{f_S}}$ represent the respective quantum circuit-depths of the quantum oracles $\mathcal{O}_{f_*}$ and $\mathcal{O}_{f_S}$, then if we are using a fully parallel approach to implementing these oracles then it may hold that $E_{\mathcal{O}_{f_*}} \approx E_{\mathcal{O}_{f_S}}$ — this can easily be seen to occur in the case where f_* and f_S are derived from a constraint-based decomposition (see Definition 2.24) and we evaluate each constraint in parallel. If we are considering circuit-depth as the sole metric in this scenario, then the *STO* technique will negatively impact upon circuit depth.

One potential countermeasure for this is trading off some of the advantage in circuit-size/circuit-width for circuit-depth via a fixing strategy and running instances in parallel. In this scenario, we have multiple quantum computers and run parallel quantum search procedures upon the problem, with k out of n bits fixed. This leads to a scenario where we are running 2^k algorithms in parallel on separate quantum computers, but each at a much smaller cost. Unfortunately it is well-known [Zal99] that Grover's algorithm at least is not embarassingly parallel procedure, in that this fixing strategy would result in a speedup of approximately $2^{k/2}$ but also increase the total number of quantum gates that we must execute by approximately $2^{k/n}$. This can easily be seen as if the circuit-size of the quantum oracle is fixed at C and the circuit-depth is fixed at D, then the total number of gates we must execute will be

$$2^k \cdot 2^{\frac{n-k}{2}} \cdot C = 2^{\frac{k}{2}} \cdot 2^{\frac{n}{2}} \cdot C \tag{6.26}$$

whilst the individual circuit-depth will be

$$2^{\frac{n-k}{2}} \cdot D. \tag{6.27}$$

We note that this fixing strategy may negate the need to use the *STO* approach, as if the number of solutions $M_S = |f_S^{-1}(1)|$ is small then fixing bits of the search-space may reduce the problem to that of a single-target search problem where we simply use the cheaper quantum oracle $\mathcal{O}_{f_S}$ with Grover's algorithm and avoid the constant increase in costs associated with using the quantum oracle $\mathcal{O}_{f_s}$ with *STO* approach.

6.3 Applications of *STO* to quantum resource estimation

We now proceed to examine applications of our approach to obtain new quantum resource estimations for solving both instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ (see Section 1.3) and the AES key-search problem (see Section 1.4). The quantum oracles and evaluations for these problems were previously discussed respectively in Sections 2.5.2 and 2.6. In this section we examine several applications of the modified *STO* approach given in Theorem 6.4 to existing quantum resource estimates in cryptanalysis, specifically cryptanalysis via direct quantum search of the $\mathcal{MQ}$ problem over $\mathbb{F}_2$ (see Section 1.3) and the key-search problem applied to the AES (see Section 1.4).

Both are instances of the single-target preimage search problem (see Definition 1.2) defined by $h : \{0,1\}^n \longrightarrow \{0,1\}^m$, $Y_h = \{y_h\} \subseteq \{0,1\}^m$ and the assumption that $M_h = |h^{-1}(y_h)| = 1$, which in turn defines the boolean indicator $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ for the unstructured search problem, where $M_\chi = |\chi^{-1}(1)|$. Both problems additionally exhibit a constraint-based decomposition so that there exist k non-trivial boolean functions $\chi_1, \ldots, \chi_k : \{0,1\}^n \longrightarrow \{0,1\}$ with the property

$$\chi(x) \mapsto \chi_1(x) \wedge \cdots \wedge \chi_k(x). \tag{6.28}$$

In relationship to the *STO* problem (see Definition 6.1) and Theorem 6.4, we will wish to define the functions $f_S, f_* : \{0,1\}^n \longrightarrow \{0,1\}$ in relationship to this decomposition. If we induce the subscript $0 < r \leq k$ to the function f_S, then we can define the function $f_{S_r} : \{0,1\}^n \longrightarrow \{0,1\}$

$$f_{S_r}(x) \mapsto \chi_{i_1}(x) \wedge \cdots \wedge \chi_{i_r}(x), \tag{6.29}$$

where $i_1, \ldots, i_r \in \{1, \ldots, k\}$ and are unique. In this case, we have that $f_* = f_{S_k}$ and $f_S = f_{S_r}$ for some chosen value of r. In the general case it will hold that $\mathcal{E}_{\chi_i}$ may be very different when $i \neq j$, but in both the cryptanalysis of the $\mathcal{MQ}$ problem over $\mathbb{F}_2$ and the key-search problem for AES-$\{128, 192, 256\}$ it will hold that $\mathcal{E}_{\chi_i} \approx \mathcal{E}_{\chi_j}$.

Given the cost framework we have introduced in Chapters 2 and 3 it will prove a relatively simple task to compute the circuit-complexity and logical qubit requirements for these scenarios. The reason why we require our modification to the *STO* algorithm given in Theorem 6.4 as opposed to simply using the solution by Kimmel et al given in Theorem 6.2 is because we cannot exactly know $M_{S_r} = |f_{S_r}^{-1}(1)|$.

6.3.1 On the probability distributions of pseudorandom functions

We will use *Markov's inequality* for purposes of computing M'_S such that $\Pr[M_S < M'_S] \geq 1 - \epsilon$.

Theorem 6.5 (Markov's inequality [GS01])**.**
If X is any positive valued random variable such that $\mathbb{E}[X] < \infty$ and $a > 0$, then

$$\Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}. \tag{6.30}$$

Corollary 6.6 (Tail bound via Markov's inequality)**.**
If X is any positive valued random variable such that $\mathbb{E}[X] < \infty$ and $a > 0$, then

$$\Pr[X \geq \frac{\mathbb{E}[X]}{a}] \leq a. \tag{6.31}$$

Proof. Substitution of $a = \frac{\mathbb{E}[X]}{a}$ in Markov's inequality. □

To ensure that the condition $\Pr[M_S \geq M'_S] \leq \epsilon$ holds for a chosen value of ϵ, we therefore simply need to take $M'_S = \frac{\mathbb{E}[X]}{\epsilon}$. With M'_S and ϵ fixed, the only parameter left to consider in Theorem 6.4 is therefore $0 \leq t \leq n$ and by numerical simulation we can compute the success probability and costs involved for various choices of t by the given formulae.

6.3.1.1 The $\mathcal{MQ}$ problem

For an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$, the respective cost of the quantum oracles $\mathcal{O}_{f_*}$ and $\mathcal{O}_{f_S}$ are therefore simply the cost of quantum oracles for an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ and a quantum oracle for an instance of $\mathcal{MQ}(\mathbb{F}_2, n, r)$ for $0 < r < m$. Explicitly, $f_* = f_{S_m}$ and $f_S = f_{S_r}$, where

$$f_{S_r}(x_1 \dots x_n) \mapsto \left(f^{(1)}(x_1, \dots, x_n) \stackrel{?}{=} 0\right) \wedge \dots \wedge \left(f^{(r)}(x_1, \dots, x_n) \stackrel{?}{=} 0\right). \tag{6.32}$$

For a random instance $\mathcal{MQ}(\mathbb{F}_2, n, m)$ of the $\mathcal{MQ}$ problem over $\mathbb{F}_2$ we expect each equation to act as a pseudorandom function, in that for a uniformly chosen $x_1 \dots x_n \in \{0,1\}^n$, we expect $f^{(k)}(x_1, \dots, x_n)$ to evaluate to 0 or 1 with equal probability. The probability of whether r equations are satisfied by a given $x_1 \dots x_n \in \{0,1\}^n$ is then the product of r independent Bernoulli trials, giving a probability of 2^{-r} that $f^{(1)}(x_1, \dots, x_n) = \dots = f^{(r)}(x_1, \dots, x_n) = 0$. The number of elements $x_1 \dots x_n \in \{0,1\}^n$ that are expected to satisfy $r \leq m$ equations is therefore the expectation of the binomial distribution with 2^n trials, each of which have a success probability of 2^{-r}. This gives us that the expected number of elements that satisfy r equations is 2^{n-r}.

The cost for the $\mathcal{MQ}(\mathbb{F}_2, n, m)$ quantum bit oracle using both the low-memory and counter-based approaches described in Section 2.4 are given in Table 2.5. We will examine three scenarios

1. where $\mathcal{O}_{f_{S_m}}$ and $\mathcal{O}_{f_{S_r}}$ are both implemented via the low-qubit oracles, so that we use $n + m + 2$ qubits.

2. where $\mathcal{O}_{f_{S_m}}$ and $\mathcal{O}_{f_{S_r}}$ are both implemented via the counter-based method, so that we use $n + \lceil \log_2(m+1) \rceil + 2$ qubits.

3. where $\mathcal{O}_{f_{S_m}}$ is implemented via the counter-based method and $\mathcal{O}_{f_{S_r}}$ is implemented via the low-qubit approach, where we use an intermediate number of qubits.

We choose $\epsilon = 2^{-15}$, which gives us that $\Pr[M_S < M'_S] \geq 1 - 2^{-15} > 0.9999$, which will give us our upper-bound for the minimal success probability of our approach.

We obtain the results in Table 6.1 below by numerical simulation to find an appropriate value of $0 < t < n$ by simply increasing t until the probability of success begins to decrease. The code used to generate these results may be found in the code attached to this thesis. As is standard in this thesis, we provide the resources required in relation to the initially proposed parameters for the Gui cryptosystem [PCY+15, PCDY17a, PCDY17b] (see Section 1.3.7). We denote the cost by $2 \cdot 2^x$ to emphasise that two serial searches are required to forge a signature for Gui.

λ	$\mathcal{MQ}(\mathbb{F}_2, n, m)$	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits	Success %	ϵ	r	t	Notes
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{78.69}$	$2 \cdot 2^{74.55}$	$2 \cdot 2^{78.77}$	$2 \cdot 2^{73.40}$	$2 \cdot 2^{78.69}$	236	99.996	2^{-15}	29	111	Low-memory
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{79.68}$	$2 \cdot 2^{75.56}$	$2 \cdot 2^{79.76}$	$2 \cdot 2^{74.39}$	$2 \cdot 2^{79.68}$	126	99.996	2^{-15}	29	112	Counter-based
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{78.84}$	$2 \cdot 2^{74.70}$	$2 \cdot 2^{78.92}$	$2 \cdot 2^{73.56}$	$2 \cdot 2^{78.84}$	153	99.996	2^{-15}	34	106	Hybrid approach
80	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 117, 117)$	$2 \cdot 2^{79.89}$	$2 \cdot 2^{74.53}$	$2 \cdot 2^{79.97}$	$2 \cdot 2^{74.53}$	$2 \cdot 2^{79.89}$	236	≈ 100.00	N/A	N/A	N/A	Grover [SW16]
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{126.45}$	$2 \cdot 2^{121.52}$	$2 \cdot 2^{126.49}$	$2 \cdot 2^{120.37}$	$2 \cdot 2^{126.45}$	420	99.996	2^{-15}	33	199	Low-memory
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{127.42}$	$2 \cdot 2^{122.49}$	$2 \cdot 2^{127.46}$	$2 \cdot 2^{121.32}$	$2 \cdot 2^{127.42}$	219	99.996	2^{-15}	34	198	Counter-based
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{126.55}$	$2 \cdot 2^{121.63}$	$2 \cdot 2^{126.59}$	$2 \cdot 2^{120.47}$	$2 \cdot 2^{126.55}$	246	99.996	2^{-15}	35	197	Hybrid approach
128	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 209, 209)$	$2 \cdot 2^{128.35}$	$2 \cdot 2^{123.38}$	$2 \cdot 2^{128.34}$	$2 \cdot 2^{122.18}$	$2 \cdot 2^{128.35}$	420	≈ 100.00	N/A	N/A	N/A	Grover [SW16]
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{252.78}$	$2 \cdot 2^{246.76}$	$2 \cdot 2^{252.81}$	$2 \cdot 2^{245.60}$	$2 \cdot 2^{252.78}$	916	99.996	2^{-15}	36	444	Low-memory
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{253.76}$	$2 \cdot 2^{247.74}$	$2 \cdot 2^{253.78}$	$2 \cdot 2^{246.55}$	$2 \cdot 2^{253.76}$	468	99.996	2^{-15}	36	444	Counter-based
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{252.83}$	$2 \cdot 2^{246.81}$	$2 \cdot 2^{252.85}$	$2 \cdot 2^{245.65}$	$2 \cdot 2^{252.83}$	497	99.996	2^{-15}	38	441	Hybrid approach
256	$2 \cdot \mathcal{MQ}(\mathbb{F}_2, 457, 457)$	$2 \cdot 2^{255.69}$	$2 \cdot 2^{249.63}$	$2 \cdot 2^{255.72}$	$2 \cdot 2^{248.42}$	$2 \cdot 2^{255.69}$	916	≈ 100.00	N/A	N/A	N/A	Grover [SW16]

Table 6.1: Quantum resource estimates for instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ using our *STO* approach.

As we can see from Table 6.1, the low-memory approach to oracle design can be improved upon enough to break the proposed parameters for Gui, though the counter-based oracle for $\lambda = 80$ and $\lambda = 128$ cannot. When we replace the cheaper oracle with the low-memory design and exploit the counter-based oracle only in the more expensive oracle, we can achieve a quantum search algorithm that break all of Gui's proposed quantum resistant parameters and only uses slightly more qubits than the counter-based approach. The algorithm as it stands is superior in all metrics to Grover, with the exception of the success probability — the author believes that with finer-grained analysis and reformulation of the algorithm that this can be improved upon.

6.3.1.2 The Advanced Encryption Standard

In the case of the key-search scenario (see Section 1.4), we are given that there exists at least one solution to this search problem, which corresponds to the users key. However, as the function $h_r : \{0,1\}^k \longrightarrow \{0,1\}^{rn}$ defined by the fixed set of r different plaintexts $P_1, \ldots, P_r \in \{0,1\}^n$ and

$$h_r(x) \mapsto \mathrm{Enc}(x, P_1) \| \ldots \| \mathrm{Enc}(x, P_r) \tag{6.33}$$

is expected to act as a pseudorandom function, the probability that the element $x \in \{0,1\}^k$ has the property that $h_r(x) = C_1 \| \ldots \| C_r \in \{0,1\}^{rn}$ for a given r is 2^{-rn}. Given that we are promised one solution, the expected number of solutions is therefore $1 + (2^n - 1) \cdot 2^{-rn} = 1 + 2^{n-rn} - 2^{-rn}$ by the binomial distribution, giving us our value of $\mathbb{E}[X]$.

As before, this problem admits a constraint-based decomposition indexed by r and the respective costs for the quantum bit oracles for the AES are given in Table 2.8 for the expensive oracle $\mathcal{O}_{f_*}$ (where $r = 2$ for AES-128/192 and $r = 3$ for AES-256) and in Table 2.9 for the cheaper oracle $\mathcal{O}_{f_S}$ (where $r = 1$ for AES-128/192/256).
Using these quantum oracles, we can therefore derive the following quantum resource estimates.

AES-k/	#Clifford gates	#T gates	#Total gates	T depth	Total depth	#Qubits	Success %	ϵ	t	Quantum algorithm/oracle type
128	$2^{85.71}$	$2^{85.32}$	$2^{86.53}$	$2^{81.02}$	$2^{82.39}$	1969	≈ 100.00	2^{-30}	50	*STO* (Theorem 6.4)/direct-evaluation
128	$2^{85.71}$	$2^{85.32}$	$2^{86.53}$	$2^{81.02}$	$2^{82.39}$	988	≈ 100.00	2^{-30}	50	*STO* (Theorem 6.4)/counter-based
128	$2^{86.05}$	$2^{85.67}$	$2^{86.87}$	$2^{80.37}$	$2^{81.73}$	1969	≈ 100.00	N/A	N/A	Grover/direct-evaluation [GLRS16]
128	$2^{86.64}$	$2^{86.23}$	$2^{87.45}$	$2^{81.92}$	$2^{83.07}$	988	≈ 100.00	N/A	N/A	Grover/counter-based [GLRS16]
192	$2^{117.89}$	$2^{117.89}$	$2^{118.89}$	$2^{112.85}$	$2^{114.0}$	2225	≈ 100.00	2^{-30}	120	*STO* (Theorem 6.4)/direct-evaluation
192	$2^{117.89}$	$2^{117.89}$	$2^{118.89}$	$2^{112.85}$	$2^{114.0}$	1340	≈ 100.00	2^{-30}	120	*STO* (Theorem 6.4)/counter-based
192	$2^{118.23}$	$2^{118.23}$	$2^{119.23}$	$2^{112.2}$	$2^{113.34}$	2225	≈ 100.00	N/A	N/A	Grover/direct-evaluation [GLRS16]
192	$2^{118.82}$	$2^{118.44}$	$2^{119.64}$	$2^{113.63}$	$2^{114.89}$	1340	≈ 100.00	N/A	N/A	Grover/counter-based [GLRS16]
256	$2^{150.21}$	$2^{149.83}$	$2^{151.03}$	$2^{145.28}$	$2^{146.42}$	4009	≈ 100.00	2^{-30}	172	*STO* (Theorem 6.4)/direct-evaluation
256	$2^{150.21}$	$2^{149.83}$	$2^{151.03}$	$2^{145.28}$	$2^{146.42}$	1340	≈ 100.00	2^{-30}	172	*STO* (Theorem 6.4)/counter-based
256	$2^{151.14}$	$2^{150.76}$	$2^{151.96}$	$2^{144.64}$	$2^{145.78}$	4009	≈ 100.00	N/A	N/A	Grover/direct-evaluation [GLRS16]
256	$2^{151.88}$	$2^{151.5}$	$2^{152.7}$	$2^{146.89}$	$2^{148.04}$	1340	≈ 100.00	N/A	N/A	Grover/counter-based [GLRS16]

Table 6.2: Quantum resource estimates for the AES key-search problem via the *STO* approach.

As we can see from Table 6.2 above, we have that we can achieve a near 100% success probability, using fewer qubits. The only metric that suffers is the quantum circuit-depth. As is plain, there is no benefit in using the direct-evaluation strategy for the secondary quantum oracle over the counter-based approach — the negligible increase in depth is absorbed by the depth contributed by the first less expensive quantum oracle $\mathcal{O}_{f_S}$ consisting of one AES-k circuit being evaluated and tested against a single ciphertext.

There is marked notice on quantum circuit-depth when comparing Grover with a direct parallel evaluation as used in [GLRS16] against the *STO* approach, we can recover some of this circuit-depth by the discussion in Section 6.2.1, using a fixing strategy and parallel evaluation to trade circuit-size for circuit-depth whilst keeping the number of qubits fixed.

6.4 Summary and future work

We have begun work on an approach to lower the projected costs of executing quantum algorithms by modifying the original solution to the *STO* problem [KYYLHH15] to take into account real-world factors, such as lack of knowledge concerning the exact value of the number of elements marked by the cheaper oracle and minor advantages of relaxing the conditions so that the algorithm uses amplification amplification instead of *exact* amplitude amplification.

There is the possibility to extend this work in several directions. In terms of the algorithm in general, whilst the approach is valid it may prove slightly advantageous — either in terms of performance or analysis — to reformulate this algorithm or (in the context of Theorem 6.4) choose slightly different values of k_1 and k_2. In particular, the author believes that Theorem 6.4 might be reformulated to provide a worst-case guarantee on its success probability that takes into account the case when $M_S \geq M'_S$. A thorough treatment of exactly when Theorem 6.4 provides an average-case advantage over the original *STO* algorithm is also missing. It is clear that for AES-128 the unmodified ratio $M_S : M'_S$ may be relatively large, but for AES$-192/256$ the original *STO* algorithm (relaxed to use amplitude amplification) using an expectation derived from the binomial distribtion should suffice in the average-case.

A tighter analysis of the discrete probability distribution for $\Pr[M_S = X]$ for either the problems we have examined or in general is also in order. In particular, the Markov inequality is a very weak bound and as the probability distribution we study for both cryptographic problems we study is known to be the binomial distribution, this work would benefit by examination how the Chernoff inequality applies. We note again that if other probability distributions are known for alternate problems, then there may be more efficient quantum search methods [Mon10] that can be used. An examination of the performance of this algorithm used both in comparison to, and in conjunction with, a fixing strategy is also an open problem.

We have only examined two simple problems in cryptanalysis. The author believes that many problems will benefit from the adapted STO approach and it would be particularly interesting to apply this problem to the case where the cost of the quantum evaluation for each constraint $E_{\mathcal{E}_{\chi_i}}$ and the expected size of the preimages $M_{\chi_i} = |\chi_i^{-1}(1)|$ are very different when $i \neq j$.

Chapter 7

Conclusions

> *The first principle is that you must not fool yourself and you are the easiest person to fool. So you have to be very careful about that. After you've not fooled yourself, its easy not to fool other scientists. You just have to be honest in a conventional way after that.*
>
> - Richard Feynman, *1974 Caltech Graduation address [Fey74]*

In this final chapter we briefly reflect on the results of this thesis, lessons learnt, the author's personal thoughts on the impact of this work and future extensions. The field of cryptographic research is by nature one that crosses many disciplines and this thesis has examined the gap in knowledge between those cryptographic and quantum algorithm design communities.

At the time of writing, the cryptographic community is currently designing new cryptographic primitives that will be (hopefully) resistant against attacks by quantum computers. Part of this process is the important topic of choosing parameters for these cryptosystems, which has a crucial impact upon the efficiency and long-term security of these schemes. As we have seen, quantum resistant parameters were previously derived for the Gui cryptosystem [PCY$^+$15, PCDY17a, PCDY17b] based upon the impact of Grover's algorithm based upon using the best known quantum attack on instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ to the cryptographic community, that being Schwabe and Westerbaan's quantum bit oracle in conjunction with Grover's algorithm. The focus of this thesis was to develop a body of evidence that the current practice of extrapolating cryptographic parameters from existing, unoptimised quantum algorithms is dangerous in terms of long-term security.

The ENTS approach was the starting work for this thesis, providing the general theme of what we can do to balance the costs involved with Grover's algorithm when there exists structure we can exploit to artificially introduce new solutions into the search domain. This was then generalised into the GENTS framework, with an examination of how preprocessing could be applied to GENTS and how, moving away from Grover's algorithm, amplitude amplification can be used to further improve these results. Finally, we examined the *Search with Two Oracles* method [KYYLHH15] and how it could be applied to both quantum bit oracles for the $\mathcal{MQ}$ problem and the key-search problem for AES, using both large and small numbers of qubits.

Whilst ENTS, GENTS and the work on preprocessing and secondary quantum search dealt mainly with optimisation of the low-memory quantum oracle design approach, the *STO* modification focused upon how quantum search could be optimised when we have only a small number of logical qubits at our disposal, demonstrating that low-qubit quantum search is not as detrimental to total quantum circuit-size as once thought if there exists structure we can exploit with the *STO* approach.

7.1 Impact

All our methods clearly break the parameters for the Gui signature scheme [PCY+15] proposed to resist attacks by quantum computers. These parameters [PCDY17a, PCDY17b] were chosen relative to a quantum attack on instances of $\mathcal{MQ}(\mathbb{F}_2, n, m)$ by Schwabe and Westerbaan [SW16]. The choice of these parameters is particularly interesting — Gui was initially proposed and analysed with respect to analysis with classical computers[PCY+15], mentioning Grover's algorithm as an afterthought. The method proposed to counter the impact of Grover's algorithm was simply to double the number of bits — in effect discounting the cost of the implementing the quantum oracle. After Schwabe and Westerbaan's quantum bit oracle was published [SW16] new parameters were proposed which took into account the entire cost of this attack [PCDY17a, PCDY17b].

In relation to the current NIST competition [oST16a], other schemes (see MQDSS [CHR+17]) as well as Gui are also having their parameters chosen relative to the projected costs of quantum search algorithms including the overhead of the quantum oracle. On the other hand, others are being more cautious and choosing lower bounds corresponding only to the query-complexity of these attacks (see GeMSS [CFMR+17]). We emphasis that our optimisations do not break the suggested parameters for the Gui submission in the NIST competition, owing to the inclusion of the MAXDEPTH parameter, which places a constraint upon the maximum depth of any quantum circuit used in the cryptanalysis of these schemes.

The author believes that for long-term security choosing parameters relative to current implementations of quantum search using Grover's algorithm is a risky business. As we have seen in Chapter 3, Grover's algorithm [Gro96] (see Theorem 3.10) can be viewed as simply a special case of amplitude amplification [BHMT02] (see Theorem 3.6), hence naively using a special case of an algorithm can be seen as being somewhat dangerous. Whilst Grover's algorithm is well-understood by the cryptographic community, the nuances of amplitude amplification seems to be less-well known. and the basic method of taking a classical circuit, converting this to a quantum circuit and using Grover's algorithm to determine the cost of quantum search, the cryptographic community is less familiar with the nuances of amplitude amplification, alternative quantum search algorithms and optimisation methods.

The safe choice would appear to be choosing the query-complexity of Grover's algorithm as a lower-bound, effectively giving the cost of implementing the quantum oracle and diffusion step as a unit cost, so that

$$E_\chi + E_{\bar{n}} + 2E_{H^{\otimes n}} = 1. \tag{7.1}$$

This clearly protects against any minor optimisations that can be made and would mean that the optimisations in this thesis have no impact on parameter sizes. A halfway measure might be instead to choose to simply allow $E_\chi = 1$, so that the overhead is simply

$$1 + E_{\bar{n}} + E_{H^{\otimes n}}, \tag{7.2}$$

as the diffusion step must be implemented for Grover's algorithm to succeed. However, optimisations such as [AdW17] impact upon this approach. This approach also ignores the fact that we can effectively hide computational work within the quantum bit oracle if we assign it a unit cost — for instance we could define the quantum bit oracle so that it performs a classical search on $K = 2^{n/2}$ elements of the search domain $\{0,1\}^n$, implying that we have a two stage search procedure where the first quantum search procedure has an execution cost of

$$\approx \sqrt{\frac{2^n}{2^{n/2}}} = 2^{n/4} \tag{7.3}$$

whilst the subsequent search of the $K = 2^{n/2}$ elements has a cost of

$$\approx \sqrt{2^{n/2}} = 2^{n/4}, \tag{7.4}$$

implying the cost of quantum search is $\approx 2 \cdot 2^{n/4}$. One safe countermeasure for this approach might be to allow the cost of the quantum oracle to be a unit cost if it only recognises a single target in the search domain. The discussion is clearly nuanced and the benefits and long-term security risks is something that must be decided by the community.

Finally, our results on the cryptanalysis of the AES key-search problem via *STO* impact upon the NIST competition directly, as the call for proposals [oST16b] explicitly states that for submissions

> "Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 128/192/256-bit key (eg.AES-128/192/256)."

Our work on cryptanalysis with small numbers of qubits via *STO* may also mean that the threat that quantum computers pose may be closer than previously thought, as whilst the number of qubits available on a quantum computer increase each year, logical qubits will most likely always be a scarce resource.

7.2 Caveats

None of the techniques in this thesis even approach removing the overhead of the quantum phase oracles with Grover's algorithm. We make modest gains, but ones that are enough to shave a few bits off the security parameters. The author believes that there is future work to be done in finding interesting optimisations to existing quantum search routines for specific problems, as many proofs and assumptions are based upon treating the quantum phase oracle as a black box.

Choosing cryptographic parameters with regards to purely the query complexity of Grover's algorithm (as was commonly done before quantum resource estimation became relatively common) is clearly the safe way to go for the single-target search case. We cannot ensure that new and improved quantum or classical algorithms are not invented, but we can protect against optimisations in this way.

Finally, we stress that the work in this thesis is relative to a specific security model involving quantum computing that the cryptographic community uses. The real cost of running any quantum search routine is expected to be vastly more expensive [ADMG+16], as it will involve quantum error-correction schemes, which imply an additional polynomial overhead over the quantum resource estimates contain in this thesis.

7.3 Future work

In terms of future work, the ENTS and GENTS framework in Chapter 4 and examples demonstrate that the balancing of costs is an important real-world (at least in this model) factor for designing quantum search algorithms — we hope that this message is absorbed by the community and that this becomes a general heuristic towards the design of quantum algorithms. It would be interesting if there were application specific ENTS that were not similar to their implementation in the GENTS framework. An example of ENTS which could be implemented where GENTS could not be or an example where ENTS provides some advantage over a nested approach to quantum search would also be interesting to find.

The quantum extension of GENTS in Chapter 5 could additionally benefit from a refined examination of both the extension to a recursive application of this quantum algorithm and optimisation. As with ENTS and GENTS, examining more applications — particularly hash functions via the Merkle-Damgård construction technique is of interest to see if there are gains.

The *STO* algorithm as given in Chapter 6 requires refinement, in terms of potentially reformulating the algorithm to allow clearer analysis and in obtaining better lower bounds for the probability of success. There is also the question of extending this to multi-target preimage search, which should yield advantageous results.

The application of these techniques to problems that impact upon industry is an open question that the author would like to investigate. Finally, whether these optimisations transfer favourably to model of quantum computing that takes into account quantum error-correction would be of immense interest to the author.

Bibliography

[Ada92] D. Adams. *Mostly Harmless*. William Heinemann, 1992.

[ADMG+16] Matthew Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent, and John Schanck. Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. In *Selected Areas in Cryptography-SAC 2016*. Springer, 2016.

[AdW17] S Arunachalam and R de Wolf. Optimizing the number of gates in quantum search. *Quantum Information & Computation*, 17(3&4):251–261, 2017.

[AFI+04] Gwénolé Ars, Jean-Charles Faugere, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison between xl and gröbner basis algorithms. *Advances in Cryptology-ASIACRYPT 2004*, pages 157–167, 2004.

[Ahl14] John Ahlgren. The probability distribution for draws until first success without replacement. *arXiv preprint arXiv:1404.1161*, 2014.

[AMM14] Matthew Amy, Dmitri Maslov, and Michele Mosca. Polynomial-time T-depth optimization of Clifford+ T circuits via matroid partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 33(10):1476–1489, 2014.

[AMMR13] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, 2013.

[Bac05] Francis Bacon. *Of the Proficience and Advancement of Learning: Divine and Human*. Oxford University, 1605.

[Bar09a] Gregory Bard. *Algebraic cryptanalysis*. Springer Science & Business Media, 2009.

[Bar09b] Gregory V Bard. The block cipher keeloq and algebraic attacks. In *Algebraic Cryptanalysis*, pages 9–16. Springer, 2009.

[BBC+95] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical review A*, 52(5):3457, 1995.

[BBHT96] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *arXiv quant-ph/9605034*, 1996.

[BCC+10] Charles Bouillaguet, Hsieh-Chung Chen, Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, Adi Shamir, and Bo-Yin Yang. Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In *Int. Workshop on Cryptographic Hardware and Embedded Systems*, pages 203–218. Springer, 2010.

[BCC+13] Charles Bouillaguet, Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, and Bo-Yin Yang. Fast exhaustive search for quadratic systems in $\mathbb{F}_2$ on FPGAs. In *International Conference on Selected Areas in Cryptography*, pages 205–222. Springer, 2013.

[BFSS13] Magali Bardet, Jean-Charles Faugére, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the complexity of solving quadratic boolean systems. *Journal of Complexity*, 29(1):53–75, 2013.

[BHMT02] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum counting. In *International Colloquium on Automata, Languages, and Programming*, pages 820–831. Springer, 1998.

[BY18] Daniel J Bernstein and Bo-Yin Yang. Asymptotically faster quantum algorithms to solve multivariate quadratic equations. In *International Conference on Post-Quantum Cryptography*, pages 487–506. Springer, 2018.

[CBW08] Nicolas T Courtois, Gregory V Bard, and David Wagner. Algebraic and slide attacks on keeloq. In *International Workshop on Fast Software Encryption*, pages 97–115. Springer, 2008.

[CFMR+17] A. Casanova, J.-C. Faugère, G. Macario-Rat, L. Patarin, J. Perret, and J. Ryckeghem. GeMSS–submission to the NIST post-quantum cryptography project., 2017.

[CG17] Yu-Ao Chen and Xiao-Shan Gao. Quantum algorithms for boolean equation solving and quantum algebraic attack on cryptosystems. *arXiv preprint arXiv:1712.06239*, 2017.

[CGW00] Nicolas J Cerf, Lov K Grover, and Colin P Williams. Nested quantum search and structured problems. *Physical Review A*, 61(3):032303, 2000.

[CHR+17] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. MQDSS—submission to the NIST post-quantum cryptography project., 2017.

[CKPS00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 392–407. Springer, 2000.

[CP03] Nicolas T Courtois and Jacques Patarin. About the XL algorithm over GF (2). In *Proc. of the 2003 RSA Cryptographers' track*, pages 141–157. Springer-Verlag, 2003.

[DC06] C Trivium De Canniere. A stream cipher construction inspired by block cipher design principles. *Information Security*, pages 171–186, 2006.

[DG10] Vivien Dubois and Nicolas Gama. The degree of regularity of hfe systems. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 557–576. Springer, 2010.

[DGS06] Jintai Ding, Jason E Gower, and Dieter S Schmidt. *Multivariate public key cryptosystems*, volume 25. Springer Science & Business Media, 2006.

[DK12] Jintai Ding and Thorsten Kleinjung. Degree of regularity for hfe minus (hfe-). *JMI: journal of math-for-industry*, 4:97–104, 2012.

[DY13] Jintai Ding and Bo-Yin Yang. Degree of regularity for hfev and hfev-. In *International Workshop on Post-Quantum Cryptography*, pages 52–66. Springer, 2013.

[Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, June 1999.

[Fau02] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero F5. In *International Symposium on Symbolic and Algebraic Computation Symposium - ISSAC 2002*, pages 75–83, Villeneuve d'Ascq, France, July 2002. ACM. Colloque avec actes et comité de lecture. internationale.

[FB09] Giordano Fusco and Eric Bach. Phase transition of multivariate polynomial systems. *Mathematical Structures in Computer Science*, 19(1):9–23, 2009.

[Fey74] Richard P Feynman. Cargo cult science. *Engineering and Science*, 37(7):10–13, 1974.

[FHK+17] Jean-Charles Faugere, Kelsey Horan, Delaram Kahrobaei, Marc Kaplan, Elham Kashefi, and Ludovic Perret. Fast quantum algorithm for solving multivariate quadratic equations. *arXiv preprint arXiv:1712.07211*, 2017.

[Flu17] Scott Fluhrer. Reassessing grovers algorithm. Technical report, Cryptology ePrint Archive, Report 2017/811, 2017. http://eprint. iacr. org , 2017.

[FMMC12] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012.

[Fra53] Gray Frank. Pulse code communication, March 17 1953. US Patent 2,632,058.

[Ghe17] Vlad Gheorghiu. Quantum resource estimationhmfev- an efficient multivariate signature scheme11. ETSI Quantum-Safe workshop 2017 (London), 2017.

[GJ79] Michael Garey and David Johnson. Computers and intractability: a guide to the theory of NP-completeness. *WH Free. Co., San Fr*, page 251, 1979.

[GKMR14] David Gosset, Vadym Kliuchnikov, Michele Mosca, and Vincent Russo. An Algorithm for the T-count. *Quantum Info. Comput.*, 14(15-16):1261–1276, November 2014.

[GKPL89] Ronald L Graham, Donald E Knuth, Oren Patashnik, and Stanley Liu. Concrete mathematics: a foundation for computer science. *Computers in Physics*, 3(5):106–107, 1989.

[GLRS16] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grover's algorithm to AES: quantum resource estimates. In *International Workshop on Post-Quantum Cryptography*, pages 29–43. Springer, 2016.

[Gro96] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proc. of the 28th annual ACM symp. on Theory of computing*, pages 212–219. ACM, 1996.

[Gro98] Lov K Grover. Quantum computers can search rapidly by using almost any transformation. *Physical Review Letters*, 80(19):4329, 1998.

[GS01] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2001.

[HHL09] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[Høy00] Peter Høyer. Arbitrary phases in quantum amplitude amplification. *Physical Review A*, 62(5):052304, 2000.

[JV17] Antoine Joux and Vanessa Vitse. A crossbred algorithm for solving boolean polynomial systems. *IACR Cryptology ePrint Archive*, 2017:372, 2017.

[KBRY15] Vadym Kliuchnikov, Alex Bocharov, Martin Roetteler, and Jon Yard. A framework for approximating qubit unitaries. *arXiv preprint arXiv:1510.03888*, 2015.

[KIT96] AY KITAEV. Quantum measurements and the abelian stabilizer problem. In *Electronic Colloq. on Computational Complexity*, 1996.

[KLM07] P. Kaye, R. LaFlamme, and M. Mosca. *An Introduction to Quantum Computing.* Oxford University Press, 2007.

[KYYLHH15] Shelby Kimmel, Cedric Yen-Yu Lin, and Lin Han-Hsuan. Oracles with costs. *10th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2015, May 20-22, 2015, Brussels, Belgium*, 44, 2015.

[LLMP93] Arjen K Lenstra, Hendrik W Lenstra, Mark S Manasse, and John M Pollard. The number field sieve. In *The development of the number field sieve*, pages 11–42. Springer, 1993.

[LPT+17] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2190–2202. SIAM, 2017.

[LW45] Clive Staples Lewis and Robert Whitfield. *The Great Divorce.* Geoffory Bles (UK), 1945.

[Mas16] Dmitri Maslov. Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization. *Physical Review A*, 93(2):022311, 2016.

[MG03] S. Milligan and A. Games. *The Essential Spike Milligan.* Fourth Estate, 2003.

[MKVOV96] Alfred J Menezes, Jonathan Katz, Paul C Van Oorschot, and Scott A Vanstone. Handbook of applied cryptography, 1996.

[Mon10] Ashley Montanaro. Quantum search with advice. In *Conference on Quantum Computation, Communication, and Cryptography*, pages 77–93. Springer, 2010.

[Mon15] Ashley Montanaro. Quantum walk speedup of backtracking algorithms. *arXiv preprint arXiv:1509.02374*, 2015.

[MR02] Sean Murphy and Matthew JB Robshaw. Essential algebraic structure within the aes. In *Annual International Cryptology Conference*, pages 1–16. Springer, 2002.

[NC10] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2010.

[oST16a] National Institute of Standards and Technology. Nist project for post-quantum cryptography standardization. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`, 2016. Accessed: 07/10/2018.

[oST16b] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process., 2016.

[PCDY17a] Albrecht Petzoldt, Ming-Shing Chen, Jintai Ding, and Bo-Yin Yang. HMFEV-an efficient multivariate signature scheme. In *International workshop on post-quantum cryptography*, pages 205–223. Springer, 2017.

[PCDY17b] Albrecht Petzoldt, Ming-Shing Chen, Jintai Ding, and Bo-Yin Yang. HMFEV-an efficient multivariate signature scheme (slides). International workshop on post-quantum cryptography, 2017.

[PCG01] Jacques Patarin, Nicolas Courtois, and Louis Goubin. Quartz, 128-bit long digital signatures. In *Cryptographers Track at the RSA Conference*, pages 282–297. Springer, 2001.

[PCY+15] Albrecht Petzoldt, Ming-Shing Chen, Bo-Yin Yang, Chengdong Tao, and Jintai Ding. Design principles for HFEv-based multivariate signature schemes. In *Int. Conference on the Theory and Application of Cryptology and Information Security*, pages 311–334. Springer, 2015.

[Pri18] Benjamin Pring. Exploiting preprocessing for quantum search to break parameters for mq cryptosystems. In *Arithmetic of Finite Fields-7th International Workshop, WAIFI 2018, Revised Selected Papers.* WAIFI, 2018.

[Pub01] NIST FIPS Pub. 197: Advanced encryption standard (aes). *Federal information processing standards publication*, 197(441):0311, 2001.

[Rei84] Rob Reiner. This is spinal tap, 1984.

[RNSL17] Martin Roetteler, Michael Naehrig, Krysta M Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 241–270. Springer, 2017.

[RS14] Neil J Ross and Peter Selinger. Optimal ancilla-free clifford+ t approximation of z-rotations. *arXiv preprint arXiv:1403.2975*, 2014.

[RS15] Martin Roetteler and Rainer Steinwandt. A note on quantum related-key attacks. *Information Processing Letters*, 115(1):40–44, 2015.

[Sel13] Peter Selinger. Quantum circuits of T-depth one. *Phys. Rev. A*, 87(4):042302, 2013.

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[Sho99] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[SW16] Peter Schwabe and Bas Westerbaan. Solving binary $\mathcal{MQ}$ with Grover's algorithm. In *SPACE 2016*, pages 303–322. Springer, 2016.

[TW12] Enrico Thomae and Christopher Wolf. Solving underdetermined systems of multivariate quadratic equations revisited. In *Public Key Cryptography*, volume 7293, pages 156–171. Springer, 2012.

[TWB+14] Sui-Guan Teo, Kenneth Koon-Ho Wong, Harry Bartlett, Leonie Simpson, and Ed Dawson. Algebraic analysis of trivium-like ciphers. In *Proceedings of the Twelfth Australasian Information Security Conference-Volume 149*, pages 77–81. Australian Computer Society, Inc., 2014.

[Wol05] Christopher Wolf. Multivariate quadratic polynomials in public key cryptography. *IACR Cryptology ePrint Archive*, 2005, 2005.

[WR14] Nathan Wiebe and Martin Roetteler. Quantum arithmetic and numerical analysis using Repeat-Until-Success circuits. *arXiv preprint arXiv:1406.2040*, 2014.

[YDH+15] Takanori Yasuda, Xavier Dahan, Yun-Ju Huang, Tsuyoshi Takagi, and Kouichi Sakurai. Mq challenge: Hardness evaluation of solving multivariate quadratic problems. *IACR Cryptology ePrint Archive*, 2015:275, 2015.

[Zal99] Christof Zalka. Grovers quantum searching algorithm is optimal. *Physical Review A*, 60(4):2746, 1999.